S-Edit User's Guide

Release 16.3

June 2015

© 1988-2015 Mentor Graphics Corporation
All rights reserved.

This document contains information that is proprietary to Mentor Graphics Corporation. The original recipient of this document may duplicate this document in whole or in part for internal business purposes only, provided that this entire notice appears in all copies. In duplicating any part of this document, the recipient agrees to make every reasonable effort to prevent the unauthorized use and distribution of the proprietary information.

This document is for information and instruction purposes. Mentor Graphics reserves the right to make changes in specifications and other information contained in this publication without prior notice, and the reader should, in all cases, consult Mentor Graphics to determine whether any changes have been made.

The terms and conditions governing the sale and licensing of Mentor Graphics products are set forth in written agreements between Mentor Graphics and its customers. No representation or other affirmation of fact contained in this publication shall be deemed to be a warranty or give rise to any liability of Mentor Graphics whatsoever.

MENTOR GRAPHICS MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

MENTOR GRAPHICS SHALL NOT BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATED TO THIS PUBLICATION OR THE INFORMATION CONTAINED IN IT, EVEN IF MENTOR GRAPHICS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

U.S. GOVERNMENT LICENSE RIGHTS: The software and documentation were developed entirely at private expense and are commercial computer software and commercial computer software documentation within the meaning of the applicable acquisition regulations. Accordingly, pursuant to FAR 48 CFR 12.212 and DFARS 48 CFR 227.7202, use, duplication and disclosure by or for the U.S. Government or a U.S. Government subcontractor is subject solely to the terms and conditions set forth in the license agreement provided with the software, except for provisions which are contrary to applicable mandatory federal laws.

TRADEMARKS: The trademarks, logos and service marks ("Marks") used herein are the property of Mentor Graphics Corporation or other parties. No one is permitted to use these Marks without the prior written consent of Mentor Graphics or the owner of the Mark, as applicable. The use herein of a third-party Mark is not an attempt to indicate Mentor Graphics as a source of a product, but is intended to indicate a product from, or associated with, a particular third party. A current list of Mentor Graphics' trademarks may be viewed at: www.mentor.com/trademarks.

The registered trademark Linux® is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a world-wide basis.

Mentor Graphics Corporation
8005 S.W. Boeckman Road, Wilsonville, Oregon 97070-7777
Telephone: 503.685.7000
Toll-Free Telephone: 800.592.2210
Website: www.mentor.com
SupportNet: supportnet.mentor.com/

Send Feedback on Documentation: supportnet.mentor.com/doc_feedback_form

S-Edit 16 User Guide—Contents

1 Interface and Setup 9

Launching S-Edit . . . 9
Parts of the User Interface . . . 10
Title Bar . . . 10
Menu Bar . . . 10
Menu List Filtering . . . 10
Toolbars . . . 10
Standard Toolbar . . . 11
Draw Toolbar . . . 11
Segment Toolbar . . . 11
Electrical Toolbar . . . 11
SPICE Simulation Toolbar . . . 11
Locator Toolbar . . . 11
Mouse Buttons Toolbar . . . 12
Customizing Toolbars . . . 12
Adding a Command to a Toolbar . . . 13
Adding a New Menu . . . 13
Adding a New Toolbar . . . 14
Adding a New Command Button . . . 15
Customizing a Command Button . . . 17
Adding or Editing a Keyboard Shortcut . . . 18
Menu and Toolbar Display Options . . . 19
Docked and Pinned Views . . . 20
Status Bar . . . 22
Design Area . . . 22
Navigation Tools . . . 24
Libraries Navigator . . . 25
Hierarchy Navigator . . . 26
Properties Navigator . . . 27
Find Navigator . . . 28
Command Window . . . 28
Copying and Pasting in the Command Window . . . 29
Command Window Log Files . . . 29
Setting Command Window Text Styles . . . 30
Filtering Information in the Log File . . . 32
Arranging Interface Elements . . . 33
Setup Options . . . 34
Saving and Loading Setup Options . . . 35
Project Setup Folder Location . . . 36
User Preferences Folder Location . . . 36
Setup Values by Design versus Library . . . 37
Setting Display Colors . . . 38
Grid and Unit Coordinates in S-Edit . . . 38
Major and Minor Display Grid . . . 39
Snap Grid . . . 39
Internal Units . . . 39
Setting the Size of the Display and Snap Grids . . . 40
Setting Unit Type and Scaling Display Units . . . 41
Setting Page Size . . . 42
Setting Cell Protection and Enabling Callbacks . . . 43
Window Reuse and Language Selection . . . 44

S-Edit 16 User Guide—Contents

(continued)

Mouse Wheel Options 45
Selection Behavior Options 46
Text File Update Options 47
Text File Display Options 48
Design Check Options 50
SPICE Simulation Options 50
S-Edit Documentation 50
S-Edit Product Support 51
Support Diagnostics 52

2 Creating a Project 53

Project Structure in S-Edit 53
Design 53
Library 53
Cell 53
Properties 53
View 53
View Types 54
Finding and Prioritizing View Types 56
How to Use Multiple Views 57
Scenario 1 57
Scenario 2 58
Scenario 3 58
Scenario 4 58
Scenario 5 59
Creating a Schematic Project 59
Creating a New Design 60
Project File Structure in S-Edit 60
Path Name Syntax and Delimiter Evaluation 61
Creating a New Text File 61
Environment Variable Usage in File Paths 61
Adding a Library 62
Opening a Design 62
Exclusive vs. Non-Exclusive Access to Designs 63
Opening a TCL File 65
Opening a TCL File for Editing 65
Executing a TCL File 65
Cells or Libraries Missing From a Design: _Unresolved 65
Resolving Missing Cells or Libraries 67
Closing a Design 69
Saving a Design or Library 69
Save Copy Of 69
Cell and View Operations 70
Shortcuts for Cell and View Commands 70
Creating a New View (Cell > New View or “N”) 71
Opening a View (Cell > Open View or “O”) 71
Copying a Cell 72
Copying a View 73
Instancing a Cell 74
Renaming a Cell 75

S-Edit 16 User Guide—Contents

(continued)

Renaming a View 75
Deleting a View 75
Printing a Design 76
Printing a Title Block 78
Print Preview 79

3 Navigating and Viewing a Design 80

The Work Area 80
Opening Design Windows 80
Reusing Design Windows 81
Changing Windows and Views 81
Splitting a Cell into Multiple Pages 82
Showing and Hiding Grids 82
Panning 83
Panning to a Specific Location 83
Zooming 83
Zooming with the Mouse 84
Zooming with the Mouse Wheel 84
Zooming with the Keyboard 84
Zooming to Selected Objects 84
Zooming to Show the Entire Contents of a Cell 84
The Libraries Navigator 84
The Hierarchy Navigator 87
Hierarchy Report in Text Format 89
The Find Navigator 90
Cross Probing from Schematic to Layout 94
Jumping to a Device in Layout 95
Jumping to a Net in Layout 96

4 Drawing, Selecting and Editing Objects 98

Object Types 98
Segment Types 98
90° Segments 98
L-Corner Segments 99
45° Segments 99
All Angle Segments 99
Using the Mouse to Draw 99
Drawing Tools for Annotation Graphics 100
Boxes (Draw > Box or “B”) 101
Polygons (Draw > Polygon or “P”) 101
Paths (Draw > Path) 101
Circles (Draw > Circle) 101
Labels (Draw > Label) 102
Auto-Repeat for All Types of Labels 103
Instance (Cell > Instance) 103
Drawing Tools for Electrical Objects 104
Wires (Draw > Electrical > Wire) 104
Solder Points (Draw > Electrical > Solder Point) 104
Connect/Disconnect (Draw > Electrical > Connect) 104

S-Edit 16 User Guide—Contents

(continued)

Net Caps (Draw > Electrical > Net Cap) . . . 105
Net Labels (Draw > Electrical > Net Name) . . . 105
Ports . . . 105
Drawn Properties of Objects . . . 105
Electrical Properties of Objects . . . 106
Selecting Objects . . . 106
Explicit Selection . . . 107
Selection by Clicking . . . 107
Selection by Enclosing (Selection Box or type Select - Enclose) . 107
Selection by Intersection (type Select - Intersect) . . . 107
Extend Selection . . . 107
Cycle Selection . . . 107
Select All (Edit > Select All) and Deselect All (Edit > Deselect All) 108
Implicit Selection . . . 108
Deselection . . . 108
Automatic Deselection . . . 108
Moving and Editing Objects . . . 109
Moving Operations . . . 109
Moving by a Specific Amount (Draw > Move By) . . . 109
Forcing a Move Operation Instead of an Edit (Draw > Force Move or Alt + M) . . . 109
Rotating Objects (Draw > Rotate 90 degrees or R) . . . 110
Flipping Objects (Draw > Flip) . . . 110
Copying and Duplicating Objects . . . 110
Creating Arrays Using the Duplicate Command . . . 111
Pasting Objects . . . 112
Pasting Objects to Other Applications . . . 112
Using Undo & Redo . . . 112
Edit > Undo (Ctrl + Z) . . . 112
Edit > Redo (Ctrl + Y) . . . 112
Showing and Hiding Objects by Type . . . 112
Deleting Objects . . . 112

5 Creating a Schematic 114

Elements of a Schematic View . . . 114
Instances of Symbols . . . 114
Ports . . . 114
Nets . . . 114
Properties . . . 114
Annotation Graphics . . . 114
Creating a Schematic View . . . 115
Creating a New View (Cell > New View or “N”) . . . 115
Creating Instances . . . 115
Updating Properties as You Place an Instance . . . 115
How To Instance a Cell (Cell > Instance or “I”) . . . 116
Updating the Name as You Place an Instance . . . 116
Locking and Hiding Instances . . . 117
Editing Instance Properties . . . 117
Editing Properties from the Work Area . . . 118
Editing Evaluated Properties . . . 118

S-Edit 16 User Guide—Contents

(continued)

Moving Instance Properties 118
Selecting All Instances of a Cell 119
Editing Properties on Multiple Instances 119
Changing Instances of a Cell to Instances of a Different Cell 119
Setting the Visibility of Properties on Instances 119
Making and Labeling Connections 119
Drawing Wires 120
Connections Points 120
Hot Spots 121
Creating a Connection where Wires Intersect 121
Rubberbanding and Disconnecting Wires 122
Adding Ports 122
Types of Ports 123
Drawing and Labeling Ports 123
Naming Ports 124
Labeling Nets 125
Editing Port Properties 127
Buses, Bundles and Arrays 127
Creating a Bus 127
Repeating Names and Grouping Buses 128
Expanding Buses 129
Creating an Array 129
Global Nets. 131
Global Ports 131
Global Symbols 132
Naming Global Nets 133
Effective Design with Global Nets 133
Capping Global Nets 134
Naming Net Caps 134
Checking a Design for Errors. 135
Design Check Setup. 136
Setup > Design Checks > Checks 136
Setup > Design Checks > Validation 138
Setup > Design Checks > Connections 139
Net Highlighting 140

6 Creating a Symbol 143

Elements of a Symbol View 143
Symbol Graphics 143
Labels 143
Ports 143
Properties 143
How To Create a Symbol 143
Visible and Hidden Properties 144
Creating and Updating Symbols Automatically 145
Symbol Property Types 147
System Properties 148
User Properties. 148
Default Properties 149
“Service” Sub-Properties 149

S-Edit 16 User Guide—Contents

(continued)

Evaluated Properties 151
Editing User Properties—the Properties Navigator 151
Shortcut Menu for the Properties Navigator 152
Properties Navigator Toolbar 153
Adding User Properties 153
Copying and Pasting Properties 154
Replacing a Device or Symbol Globally 154
Selecting Properties from the Layout 155
"Callbacks" for Property Values 155
Automatically Setting Properties During Cell > Instance 156
Writing TCL Functions for Callbacks 157
Port Placement 158

7 Evaluated Properties and Labels 160

Expressions as Property Values 160
Displaying Evaluated Properties 160
Evaluated Text Labels 161
S-Edit supports TCL expressions in text labels. A property on a text label called Evaluation, with value TRUE, FALSE or EXPRESSION, indicates whether the the label is to be evaluated. Here, EXPRESSION is an expression that evaluates to TRUE or FALSE. 161
Supported Operators, Functions and References 161
TCL Commands in Expressions 163
Built in TCL functions 163
Selective Evaluation: se 163
Annotate Commands 164
Cadence cdsParam(n) Labels 167
Viewing Property Values In Context 167

8 Importing and Exporting Netlists and Schematics 168

Importing a Design 168
Importing SPICE Files 168
Importing EDIF Files 169
Importing EDIF from S-Edit or ViewDraw 170
Importing EDIF from Cadence Composer 172
Importing EDIF from Mentor Design Architect 175
Importing OpenAccess 176
Importing Verilog Files 177
Importing CDF Files 179
Exporting a Design 180
Exporting SPICE Files 180
SPICE Export Properties 183
SPICE Output Examples 188
Example 1: MOSFET 188
Example 2: MOSFET with Property Substitution 189
Example 3: Conditional Output 189
Example 5: Title Block 190
Passing Subcircuit Parameters to the Originating Cell 190
Exporting Global Node Connections 190

S-Edit 16 User Guide—Contents

(continued)

Exporting EDIF Files 191
Exporting Verilog Files 193
Properties Controlling Verilog Export Behavior 194
Exporting VHDL Files 195
Properties Controlling VHDL Export Behavior 196
Exporting TPR Files 197
TPR File Format 198

9 Scripting with TCL 200

S-Edit TCL Command Help System 200
TCL Commands Available in S-Edit 201
Source Scripts 202
Executing Scripts Automatically 203
Load Order for TCL Files 205
Load Order for Libraries 205

10 Simulation and Waveform Probing 207

SPICE Simulation Settings 208
All Fields in Setup SPICE Simulation are Evaluated 208
Importing and Exporting SPICE Simulation Settings 208
General Settings 210
Simulation Outputs 211
File and Directory Names 211
External Simulator Setup 214
Netlisting Options 214
Hierarchy Priority 216
Additional SPICE Commands 218
SPICE Command Wizard 219
Online SPICE Model and Parameter Lists 220
Parameters 221
SPICE Options 221
DC Operating Point Analysis 222
Transient/Fourier Analysis 222
DC Sweep Analysis (or DC Transfer) 224
AC Analysis 226
Noise Analysis 227
Transfer Function Analysis 227
Temperature Sweep 228
Parameter Sweep 228
Corner Simulations 230
Running Simulations. 232
Probing Waveforms. 233
Using the .probe Command 233
Probing Setup 234
Probe Tools 235
Probing for Unique Instance Properties—“In Context” Values . . . 236
Displaying Schematic Annotations 237

S-Edit 16 User Guide—Contents

(continued)

Index	**238**
Credits	**246**

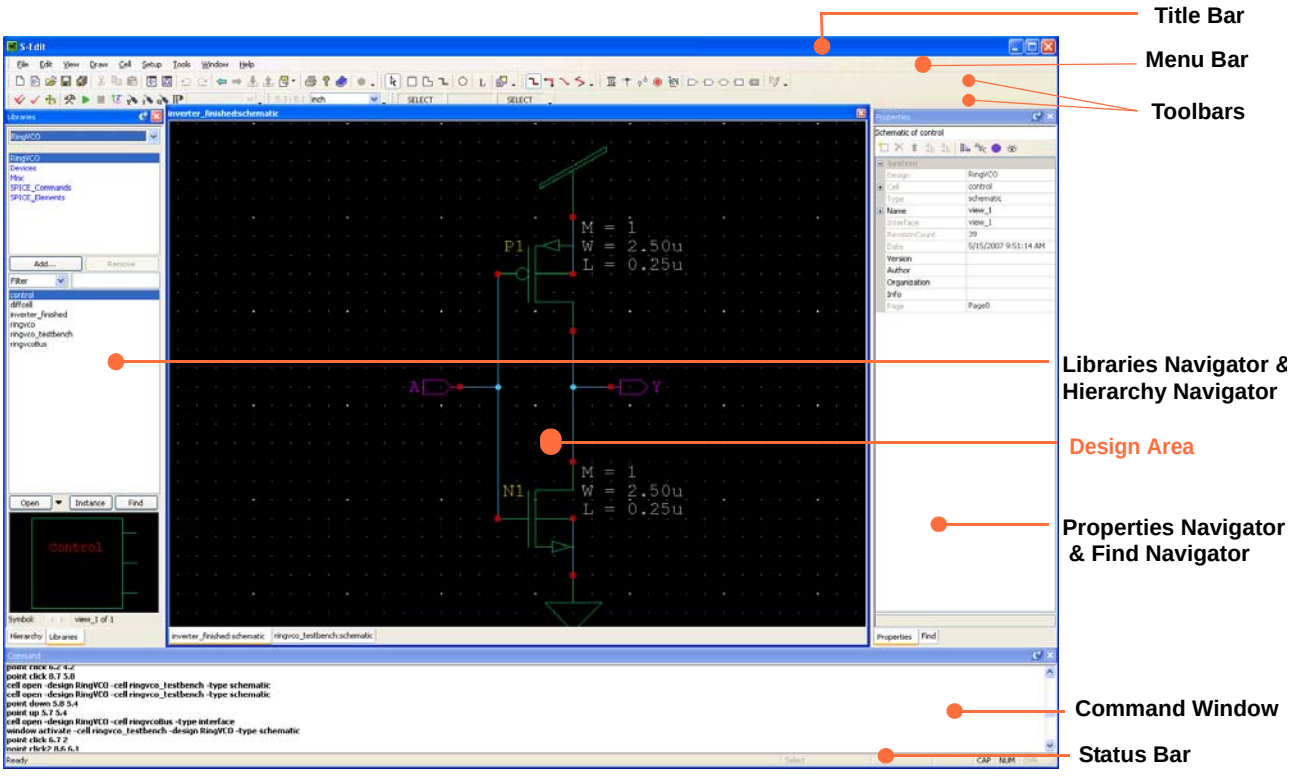
1 Interface and Setup

S-Edit is a fully hierarchical computer-aided schematic capture application for the logical design of integrated circuits. S-Edit contains integrated SPICE simulation and probing of simulation results, including voltages, currents, and noise parameters.

Launching S-Edit

To launch S-Edit, double-click on the S-Edit icon.

The user interface consists of the elements shown below. Unless you explicitly retrieve a setup file, the position, docking status and other display characteristics are saved with a design and will be restored when the design is loaded.



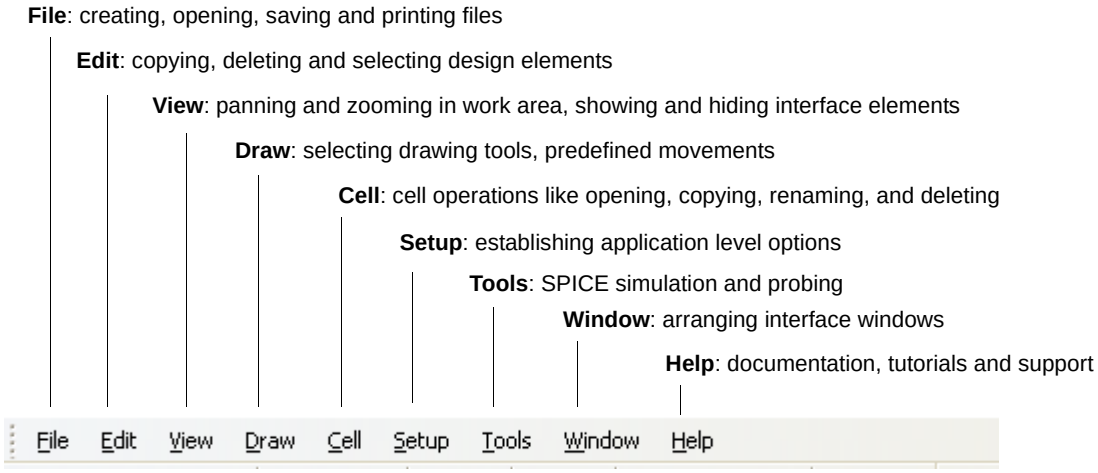
Parts of the User Interface

Title Bar

The *title bar* shows the name of the current cell and the view type (symbol, schematic, etc.).

Menu Bar

The *menu bar* contains the S-Edit menu titles. The menu displayed may vary depending on the view type that is active. See “Shortcuts for Cell and View Commands” on page 70 for the various methods S-Edit provides for executing commands.


File: creating, opening, saving and printing files
Edit: copying, deleting and selecting design elements
View: panning and zooming in work area, showing and hiding interface elements
Draw: selecting drawing tools, predefined movements
Cell: cell operations like opening, copying, renaming, and deleting
Setup: establishing application level options
Tools: SPICE simulation and probing
Window: arranging interface windows
Help: documentation, tutorials and support
File Edit View Draw Cell Setup Tools Window Help

Menu List Filtering

Most S-Edit menus and dialogs allow for filtering to speed the process of selecting from a drop-down list. So, when you enter a character, S-Edit will jump to the first list item that begins with that character. For example, typing **g** highlights the first list item beginning with that letter and filters the display to show only items that begin with **g**. Typing a **u** after the **g** highlights the first list item beginning with **gu**, and filters the display to show only items that begin with **gu**, and so on. The search procedure is case-insensitive.

Toolbars

You can display or hide individual toolbars using the **View > Toolbars** command, or by right-clicking in the toolbar region. Toolbars can be relocated and docked as you like. For added convenience, S-Edit displays a tool tip when the cursor hovers over an icon.

You can also customize your toolbars; see “Customizing Toolbars” on page 12.

Standard Toolbar

The *Standard* toolbar provides buttons for commonly used file and editing commands, as well as operations specific to S-Edit such as “View Symbol.”

Draw Toolbar

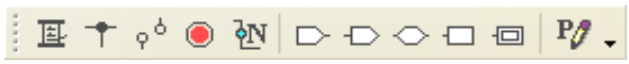
The *Draw* toolbar provides tools used to create non-electrical objects, such as rectangles, circles, and lines, for illustrating and documenting a design.

Segment Toolbar

The *Segment* toolbar *provides tools with which you limit the degree of angular freedom allowed when you are drawing wires. (See* “Drawing Tools for Electrical Objects” on page 104.)

Electrical Toolbar

The *Electrical* toolbar provides the tools used to create wires, nets, and ports, and to add properties.

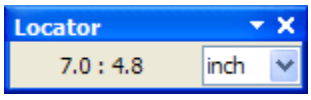


SPICE Simulation Toolbar

The *SPICE Simulation* toolbar lets you extract connectivity, select and probe nets, launch T-Spice and select evaluated properties.

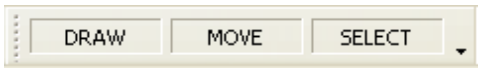
Locator Toolbar

The *Locator* toolbar displays the coordinates of the mouse cursor and allows you to quickly change the units of measurement application-wide.



Mouse Buttons Toolbar

The *Mouse Buttons* toolbar shows the current functions of the mouse buttons.

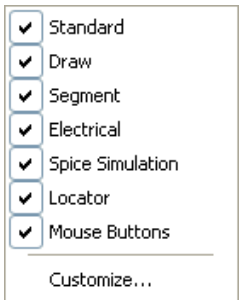


Mouse buttons vary in function according to the tool that is active. The **Shift**, **Ctrl** and **Alt** keys can further change the function. For two-button mice, the middle-button function is accessed by clicking the left and right buttons at the same time, or by pressing **Alt** while clicking the left mouse button.

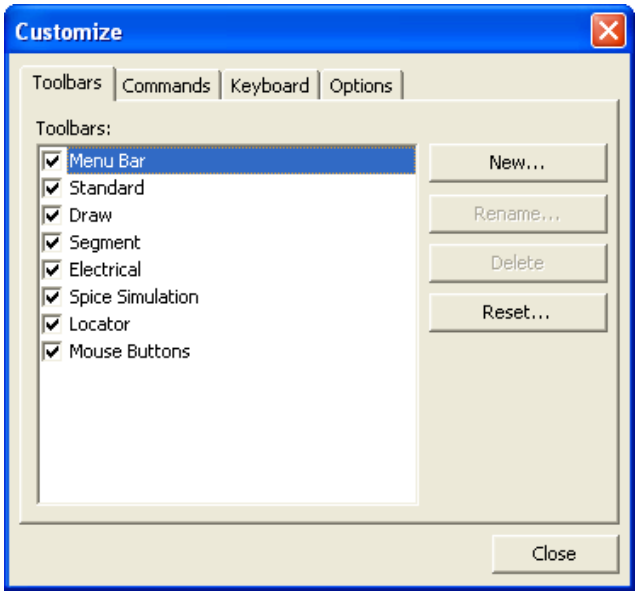
Customizing Toolbars

You can add buttons for existing commands to existing S-Edit toolbars, add entirely new toolbars, and add new buttons for entirely new commands to either new or existing toolbars.

To customize toolbars, right-click anywhere in the toolbar area and click on **Customize** in the context-sensitive menu.



This opens the **Customize** dialog, to the **Toolbars** tab. Note that in this dialog the checkmarks control only whether or not a toolbar is displayed. The buttons apply only to the toolbar that is highlighted, and will be applied even if a toolbar is not currently displayed.



All toolbars are checked, so all are displayed.

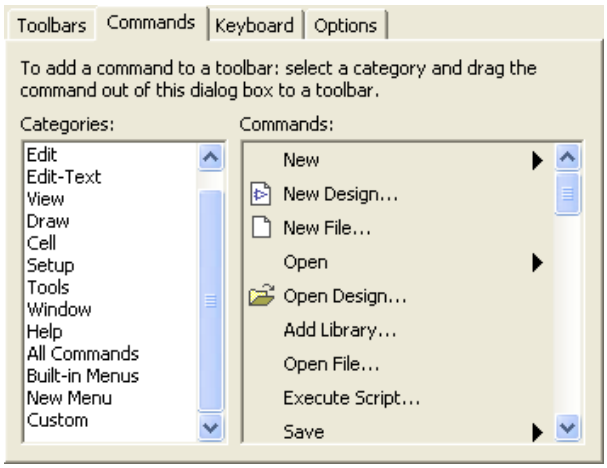
Only Menu Bar is highlighted, so any of the button actions (ex. Reset) will act only on the Menu Bar.

Reset returns an existing toolbar to the default display settings for aspects such as icon size, tooltips, etc. – see “Menu and Toolbar Display Options” on page 19 – and its original button contents.

The **New**, **Rename** and **Delete** functions apply only to custom toolbars (see “Adding a New Toolbar” on page 14.)

Adding a Command to a Toolbar

Use the **Commands** tab to add a button for an existing command to any toolbar.



[1] Right-click in the toolbar area, select **Customize** and then the **Commands** tab.

[2] Pick the desired command from the **Categories** list (or use All Commands for a complete list of available commands), then simply click-and-drag the command from the right column to the desired toolbar.

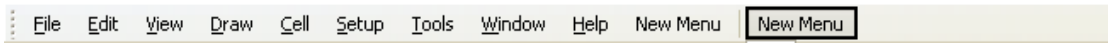
[3] S-Edit will insert a button displaying the command text, or an icon if one is already defined.

Adding a New Menu

[1] You can also use the **Commands** tab to add a new menu category to the menu bar.

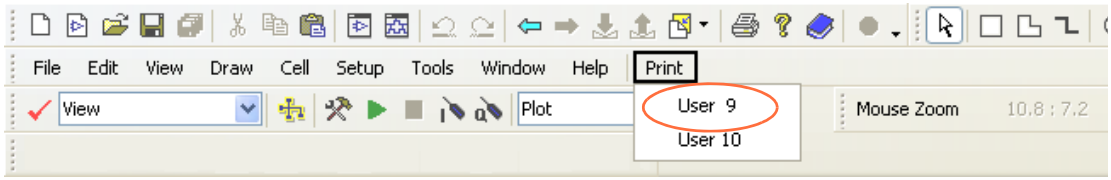
[2] In the Commands tab, scroll down to **New Menu** at the end of the **Categories** list.

[3] Click-and-drag **New Menu** from the right column to the Menu bar in the interface.



[4] Right-click on the New Menu button you have just placed to open the control menu, where you can rename it, then check **Begin a Group** to populate the menu with pull-down commands.

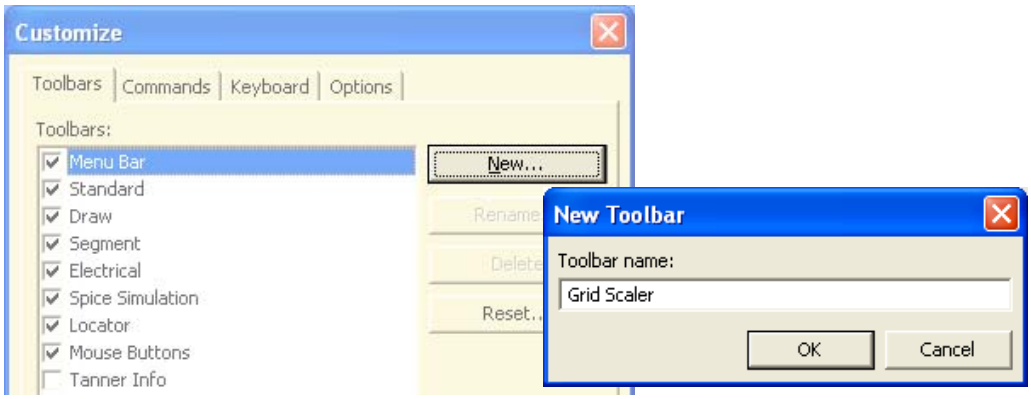
[5] Select the new menu button in the interface to open the pull-down group, then click-and-drag from the Commands tab to add the desired command(s). Make sure to drop the commands within the group area.



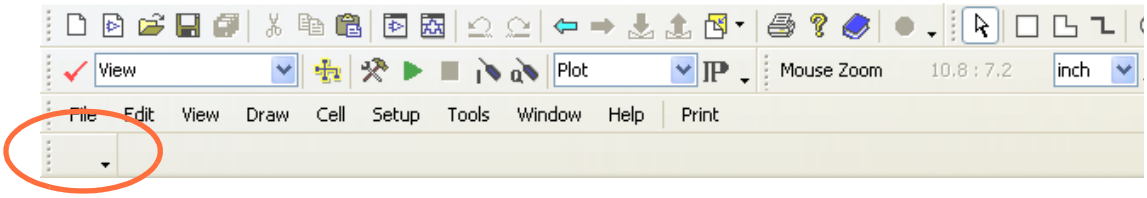
Adding a New Toolbar

[1] Right-click in the toolbar area, select **Customize** and click on the **New** button.

[2] Enter the desired name in the **New Toolbar** window and click OK to display it in the toolbar area.



[3] Note that although it has a name, the new toolbar is small and blank when first placed – you may have to search a bit to find it.



[4] If you have just added the toolbar, you can click-and-drag from the **Commands** pane to add an existing command. Otherwise, right-click in the toolbar area, select **Customize**, highlight a menu in the **Categories** pane then click-and-drag the desired command from the **Commands** pane to your new toolbar.

Adding a New Command Button

There are two important requirements for adding a new command to S-Edit. You must write and execute a TCL function to perform the desired command, and you must create a custom button for the command that has the same name as the TCL function.

As long as the TCL function is loaded into S-Edit during the current editing session, S-Edit will run the function when you press the custom button to execute the operation. Lastly, if you want a button to work in subsequent sessions, you will need to save it to an S-Edit startup folder.

Example: Adding a Button to Toggle Snap Grid Size

Please follow the steps in this example to learn how to add a new command button. In this example we will add a toolbar with two new buttons; one that doubles the size of the current snap grid and another that halves it.

[1] Enter and execute each of these TCL functions separately in the S-Edit command window. Note that they must be entered in one unbroken line:

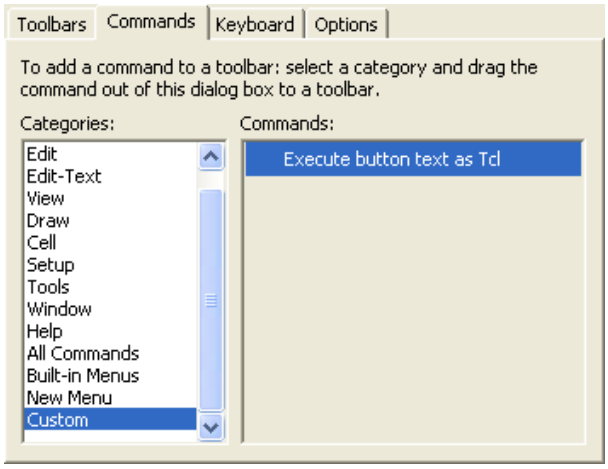
```
proc GridDouble {} { setup schematicgrid set -snapgridsize [expr 2*[setup
   schematicgrid get -snapgridsize]] -units iu }

proc GridHalf {} { setup schematicgrid set -snapgridsize [expr .5*[setup
   schematicgrid get -snapgridsize]] -units iu }
```

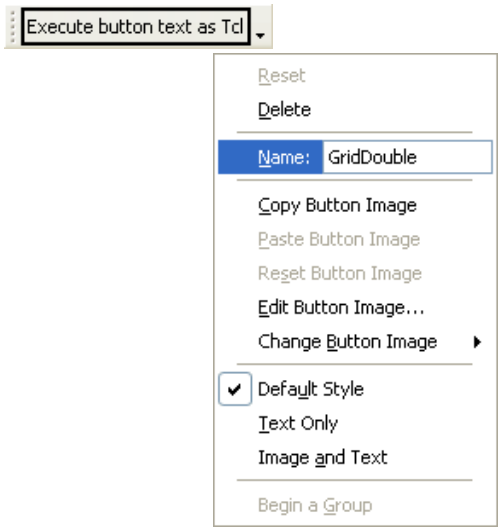
[2] Create a new toolbar named "**Custom Snap Grid**" (see "Adding a New Toolbar", above).

[3] If you have just added the toolbar you can click on the **Commands** tab, scroll down to the bottom of the list and select **Custom**. If not, right-click in the toolbar area, select **Customize**, highlight the "Custom Snap Grid" menu, click on the **Commands** tab, and scroll down to the bottom of the list and select **Custom**.

[4] Grab the text "**Execute button text as Tcl**" from the right pane and drag it to the newly created toolbar



[5] For custom buttons you must replace "Execute button text as Tcl" with the name of the TCL function you want to use. With the "Execute button text as Tcl" button highlighted, right-click and enter the function name "**GridDouble**" in the name field.



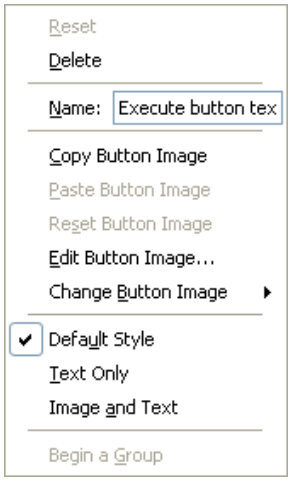
[1] Repeat step 4 to add a second "Execute button text as Tcl" button to the toolbar. Right-click on the new button and name it "**GridHalf**."

[2] With a schematic view open, launch **Setup > Technology > Schematic Grids**. Note the current snap grid size, then press GridDouble and GridHalf a few times to confirm that the buttons are working.

[3] The scripts will execute for the duration of the current session. To execute them each time S-Edit launches you must save the TCL commands, as GridHalf.tcl and GridDouble.tcl, in the S-Edit startup directory (typically C:/Documents and Settings/<*username*>/Application Data/Tanner EDA/scripts /startup for Windows XP or C:/Users/<*username*>/AppData/Roaming/Tanner EDA/ scripts/startup for Windows 7.)

Customizing a Command Button

Right-click on a toolbar button with the toolbar **Customize** dialog open to access the button controls shown below.



Reset — Returns standard buttons to their default shipping text and icon display.

Delete — Deletes the selected button.

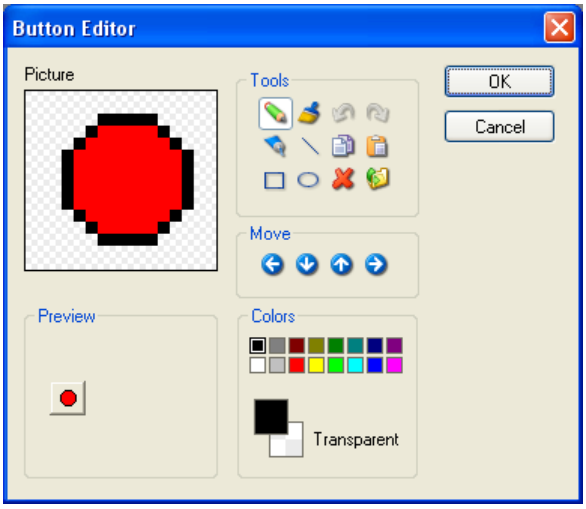
Name — Use this field to edit the button name when it is displayed as text, or to enter the name of the TCL command to issue. The tooltip will not be affected.

Copy Button Image — Copies the selected button image.

Paste Button Image — Pastes the selected button image.

Reset Button Image — Resets all changes to the button image and text.

Edit button Image — Opens the **Button Editor** where you can perform basic graphic functions.



Change Button Image Opens a small palette of clip art from which you can choose an icon.

Default Style Displays the default (image only) for the selected toolbar button.

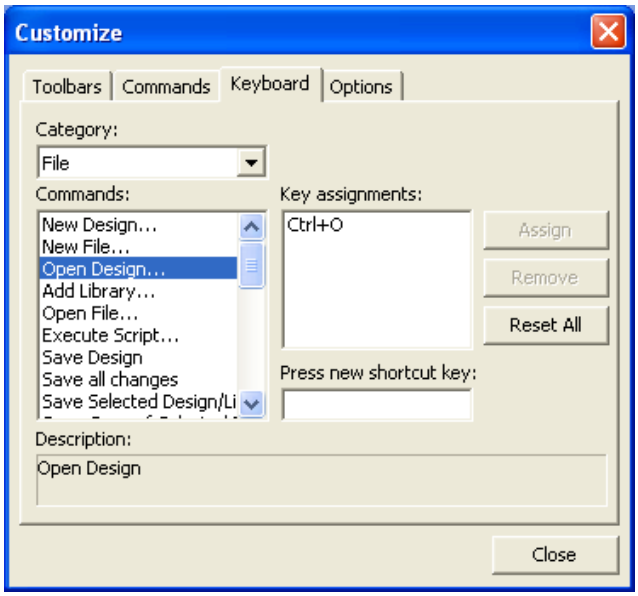
Text Only Displays just the contents of the **Name** field for the selected toolbar button.

Image and Text Displays both the icon and the text from the **Name** field for the selected toolbar button.

Begin a Group Inserts a fine line denoting a toolbar group to the left of the selected button.

Adding or Editing a Keyboard Shortcut

Customize > Keyboard lets you add or change shortcut key assignments for menu commands.



Category: Select the menu to which the command belongs.

Commands: Select the command for which you want to add or change a keyboard shortcut.

Key assignments: Displays existing key assignments. If blank, no shortcut is assigned. It is possible to have more than one shortcut for a command.

Highlight a command and click on **Remove** to remove a shortcut.

To restore the default settings click on **Reset All**.

Click on **Assign** to save a shortcut.

Press new shortcut key: Highlight a command in the **Commands** pane, then use this field to enter the key(s) that will be the shortcut.

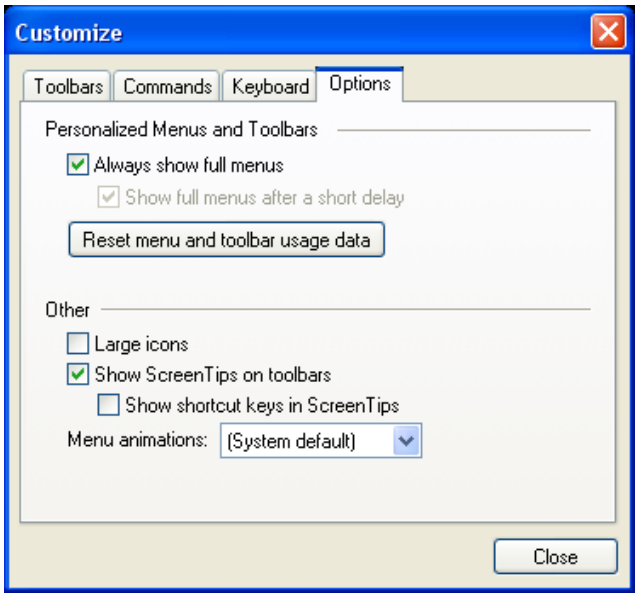
You can use any combination of the **Alt**, **Shift** and **Ctrl** keys with any of the character keys. S-Edit will warn you if your entry is already in use. (Since this field interprets any key you press literally, you cannot delete a value in this field—simply enter a different value.)

Click on **Assign** to save your shortcut.

Description: This display-only field shows the tooltip, if any, for a command.

Menu and Toolbar Display Options

Customize > Options lets you set default display settings for menus and toolbars.



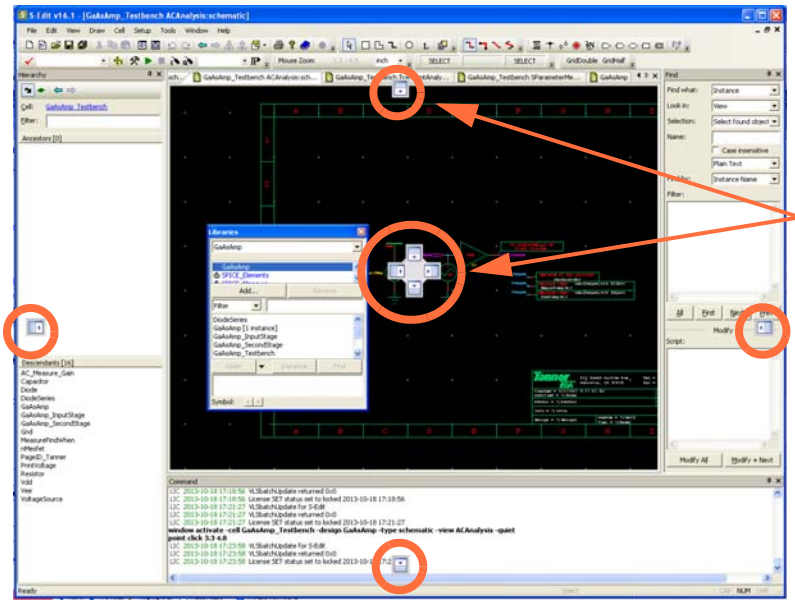
Always show full menus Menus and toolbars will automatically adjust based on how often you use commands so that only the commands you use most often are displayed. If you prefer, you can choose the **Always show full menus** option so that all commands are displayed on the menu.

Show full menus after a short delay (Not operational.)

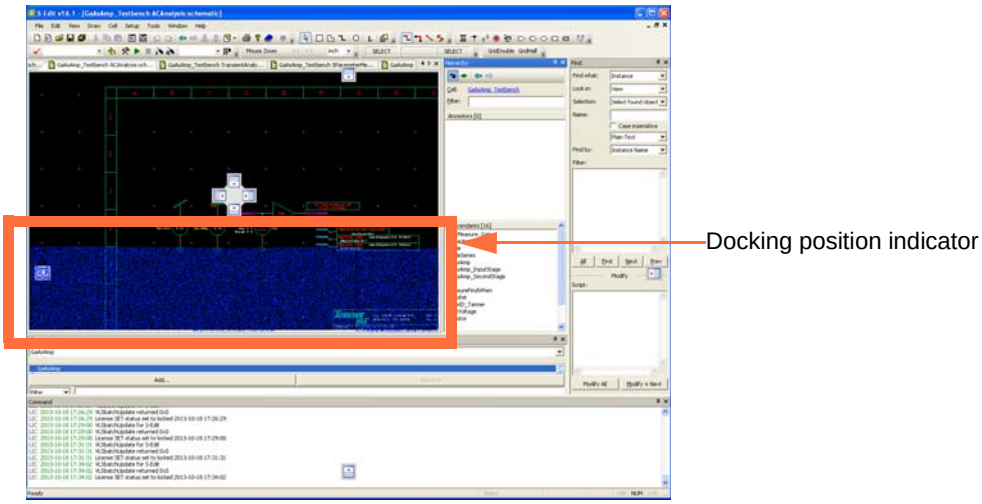
Reset menu and toolbar usage data	Deletes the record of all the commands used in S-Edit (for short menu display) and restores the default set of visible commands to the menus and toolbars. However, explicit changes you have made in the current or earlier sessions will remain.
Large icons	Check this box to display large toolbar buttons.
Show Screentips on toolbars	Check this box to display screentips (also called “tooltips”) for the toolbar buttons. (A tooltip is a brief reminder of the related command. It is displayed when your mouse hovers over a toolbar button.)
Show shortcut keys in Screentips	Check this box to add shortcut keys to the tooltip display.
Menu animations	You can choose from the list to add animation to the open menu operation.

Docked and Pinned Views

Dockable windows are interface objects that can be attached to the edge of a fixed or main window, as well as being able to float anywhere in the Windows interface. To change the location of a dockable window, click and drag it using the titlebar. This will trigger display of shaded blue indicators with arrows pointing to where the window will be docked.

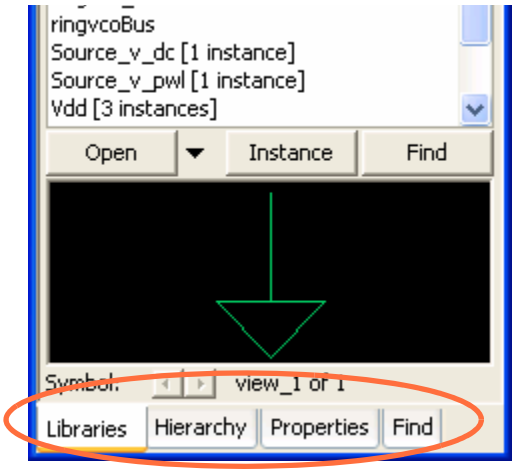


When the titlebar of a docking view is positioned over an indicator, a sheer blue rectangle appears to show where the window will be docked.



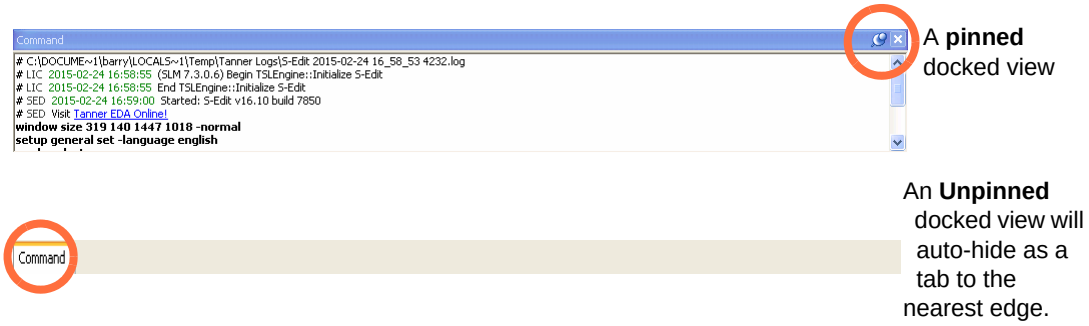
The window will dock when you release the mouse button with the titlebar over a docking indicator. If the window is not over a docking indicator when you release the mouse button, it will be placed as a floating window.You can resize a floating window by dragging any of its edges or corners. A docked window can be resized, but only from the edge opposite its docked edge.

Multiple dockable windows can be grouped together. When this is the case, only one of the windows is visible. The others are shown as tabs. Click on a tab to bring a docked window to the fore and hide the others as tabs.



Pinning A Docked View

Docked views include a “pin” feature that will automatically hide a docked window by minimizing it into tabs when not in use. When you click on the tab for a pinned view, the window will reopen in its docked state.


Command
C:\DOCUME~1\barry\LOCALS~1\Temp\Tanner Logs\S-Edit 2015-02-24 16_58_53 4232.log
LIC 2015-02-24 16:58:55 (SLM 7.3.0.6) Begin TSLEngine::Initialize S-Edit
LIC 2015-02-24 16:58:55 End TSLEngine::Initialize S-Edit
SED 2015-02-24 16:59:00 Started: S-Edit v16.10 build 7850
SED Visit Tanner EDA Online!
window size 319 140 1447 1018 -normal
setup general set -language english
A pinned docked view
Command
An Unpinned docked view will auto-hide as a tab to the nearest edge.

Status Bar

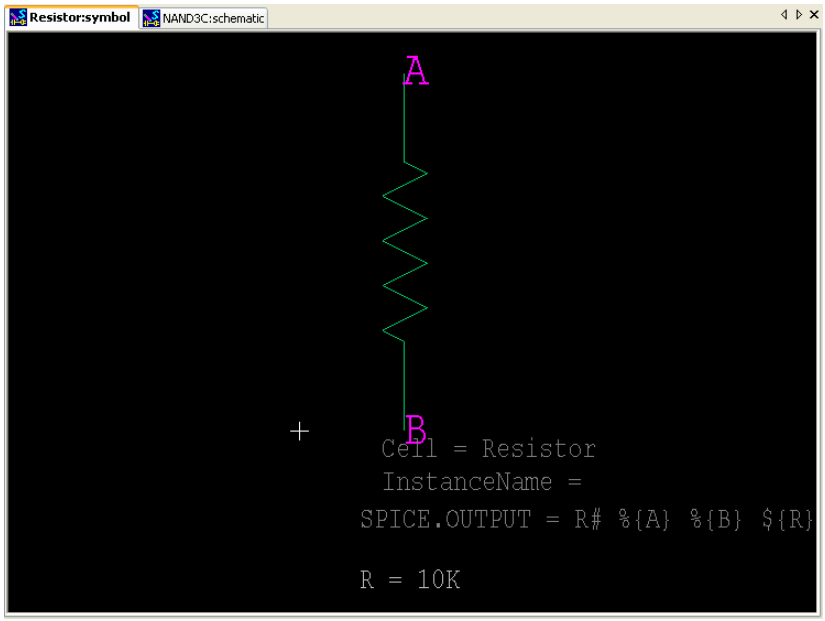
The *Status Bar* display varies with the type and number of objects selected and the tool in use. You can use **View > Status Bar** to toggle display on and off.

Item Selected	*Status Bar displays:*
single instance	The instance name and source cell
simple geometry	Width, height, area, pathlength, the number of vertices it contains and for ports, the X,Y coordinates and name of the object.
multiple instances	The number of instances selected.
multiple objects	The type and number of each object.
during design probing	The status of waveform probing operations.
during drawing operations	The drawing mode and any constraints (on the right side).

Design Area

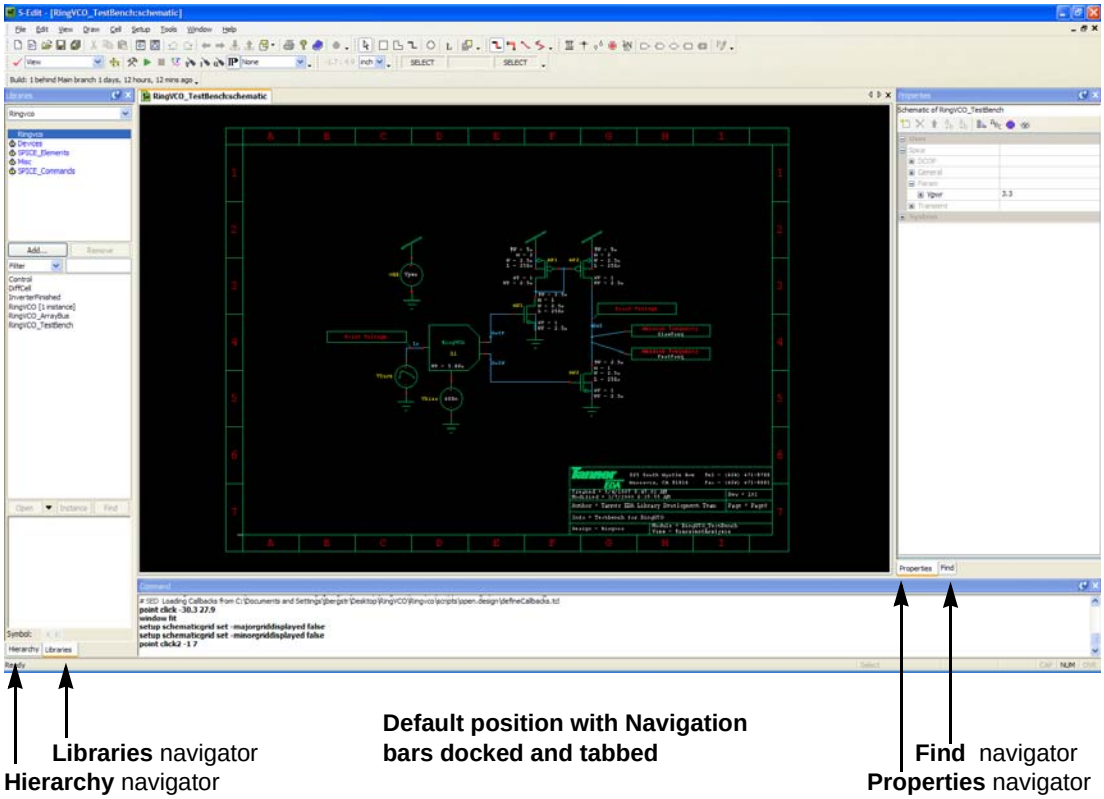
The region in Tanner tools where you create, view and edit objects is called the *Design Area*. The portion of the design area currently visible is called the *Work Area*. You can move or resize design

windows as you would in any other application window. Refer to “The Work Area” on page 80 for further information.



Navigation Tools

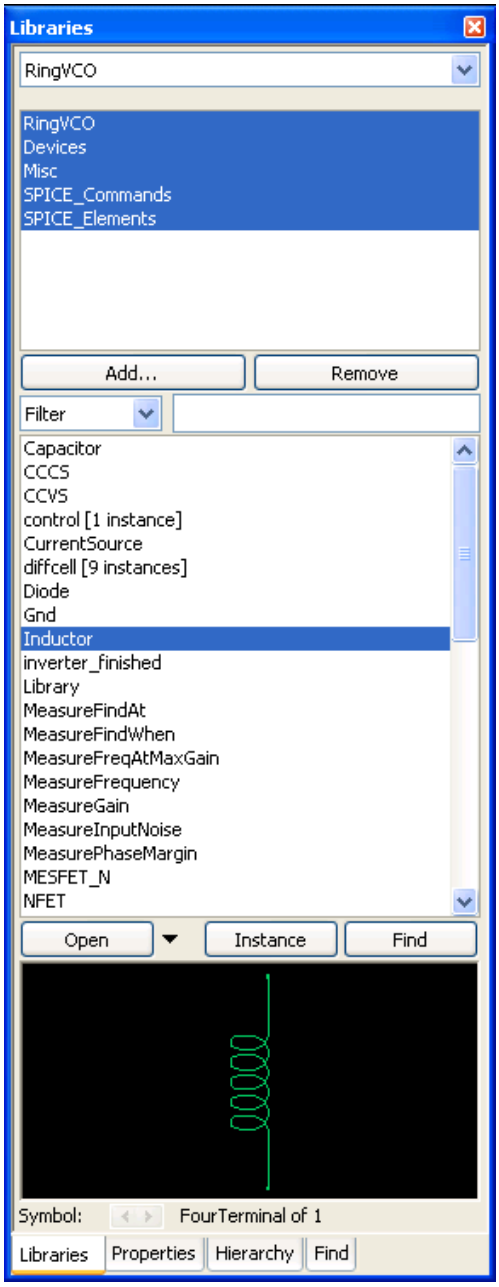
S-Edit provides several different tools for displaying and searching design libraries, cells, and cell properties. Default positioning is with the **Libraries** and **Hierarchy** navigators tabbed together on the left and the **Properties** and **Find** navigators on the right.



Each of the navigators is a dockable toolbar. They can be placed in any of the standard Windows docking locations, they can be positioned anywhere in the S-Edit interface as a floating window, they can be completely separated or grouped all together in a tabbed window.

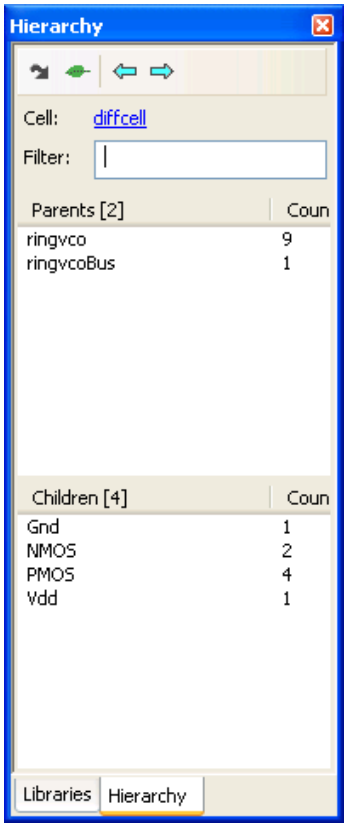
Libraries Navigator

The Libraries navigator allows you to add and remove cell libraries and also displays a list of the cells in the libraries that are highlighted. You can use this list to open, duplicate, or instance a cell, or to create a new view of a cell. See “The Libraries Navigator” on page 84 for more information.



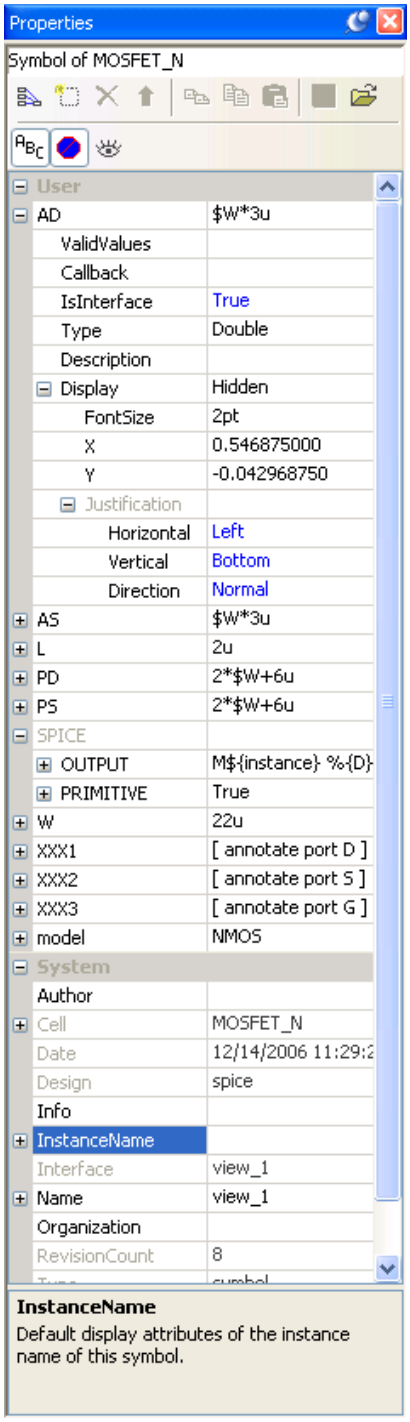
Hierarchy Navigator

The Hierarchy navigator displays hierarchical information for the active cell, including the parent and child cells, the total count of each, and the number of times they are instanced. See “The Hierarchy Navigator” on page 87.



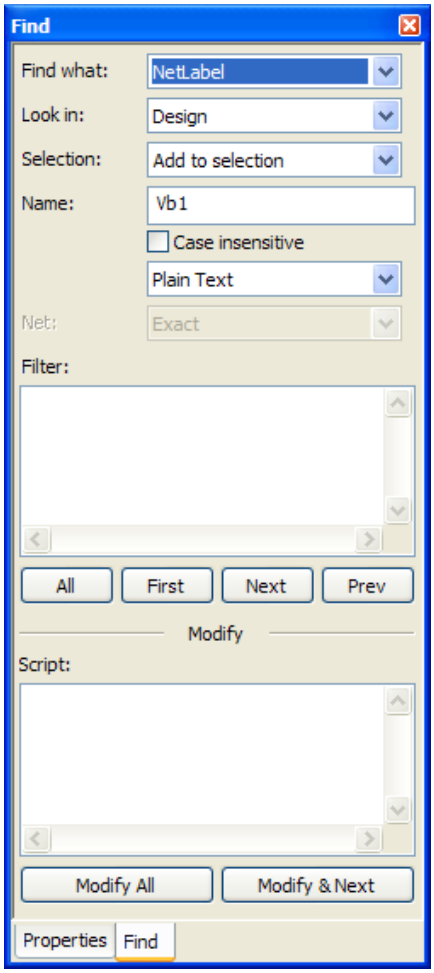
Properties Navigator

The Properties navigator provides an editable display of the characteristics of a selected object, which may be a cell, drawn geometry, port, etc. See “Editing User Properties—the Properties Navigator” on page 151 and “Editing Instance Properties” on page 117.



Find Navigator

The Find navigator provides advanced find options for any object type, with fields for TCL scripting that can be used to further define a search. See “The Find Navigator” on page 90.

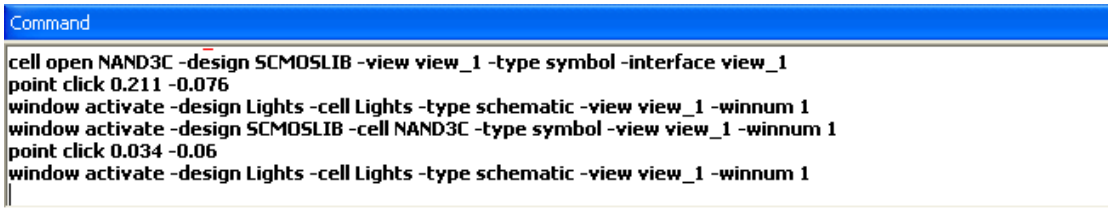


Command Window

All events that occur in the S-Edit design area are recorded in tool command language (TCL) format to a log displayed in the *Command window*. TCL files are macro-like scripts that allow you to perform or repeat operations.

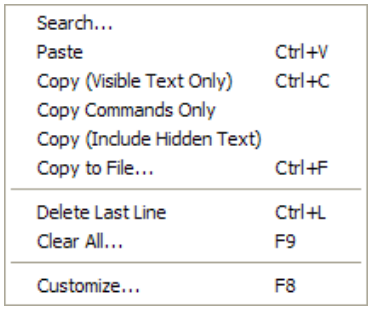
The Command window serves as both a recording and a playback device for TCL files. As such, any action or operation performed by S-Edit can be copied or replayed. Text can be typed in, copied from executed operations and then pasted back into the Command window, or written in from a saved TCL file to instantly perform the desired operations. This is especially useful for automating and simplifying difficult or repetitive tasks.

You can use **View > Activate Command Window** (shortcut backquote‘) to open the Command window if it is not open. If it is open, backquote will shift focus to it.


Command
cell open NAND3C -design SCMOSLIB -view view_1 -type symbol -interface view_1
point click 0.211 -0.076
window activate -design Lights -cell Lights -type schematic -view view_1 -winnum 1
window activate -design SCMOSLIB -cell NAND3C -type symbol -view view_1 -winnum 1
point click 0.034 -0.06
window activate -design Lights -cell Lights -type schematic -view view_1 -winnum 1

Copying and Pasting in the Command Window

A right-click in the Command window opens the menu below.



Search	Searches the contents of the command window.
Paste	Copies the contents of the clipboard to the command window.
Copy (Visible Text Only)	Copies just the text visible in the command window.
Copy Commands Only	Copies just the TCL commands from the command window.
Copy (Include Hidden Text)	Copies all formatting tags and hidden text from the command window.
Copy to File	Opens a Save As window so you can paste the contents copied from the command window into a .tcl file.
Delete Last Line	Deletes just the last line from the command window.
Clear All	Clears the entire contents of the command window.
Customize	Opens the **Customize Command Window** dialog, which is discussed in the following sections: ▪ “Command Window Log Files,” below ▪ “Setting Command Window Text Styles” on page 30 ▪ “Filtering Information in the Log File” on page 32

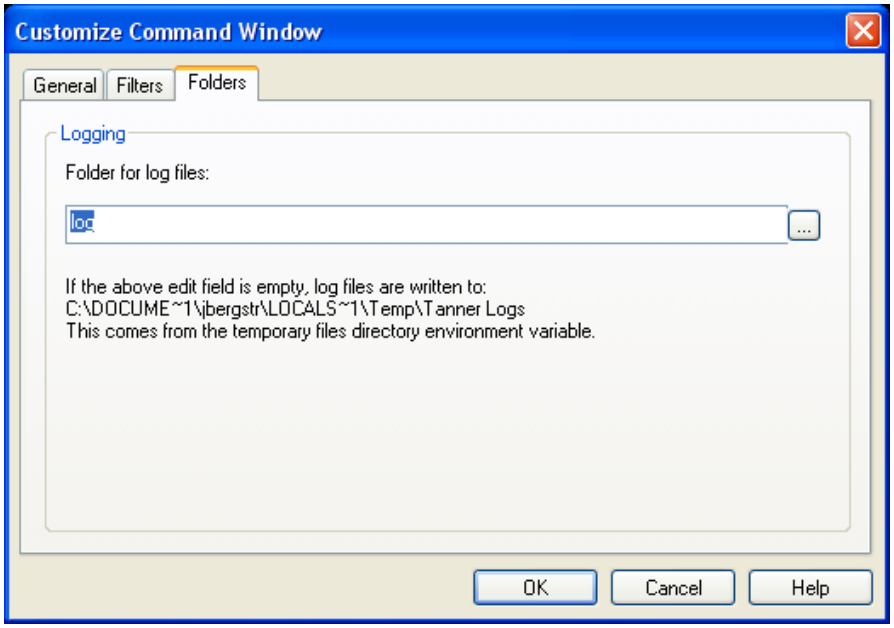
Command Window Log Files

Each time S-Edit is launched it creates a TCL log file that records each operation performed in the design windows for the entire session. The last 10,000 lines (a default value which can be changed, see

"Filtering Information in the Log File" on page 32) of this log are displayed in the Command window. Logs files are identified by the date and time the session was launched.

How to Locate Log Files

To set the directory where log files are stored, right-click in the Command window to open its context-sensitive menu. Select **Customize** (shortcut **F8**) to open the **Customize Command Window** dialog, then use the **Folders** tab to select or create a storage directory.



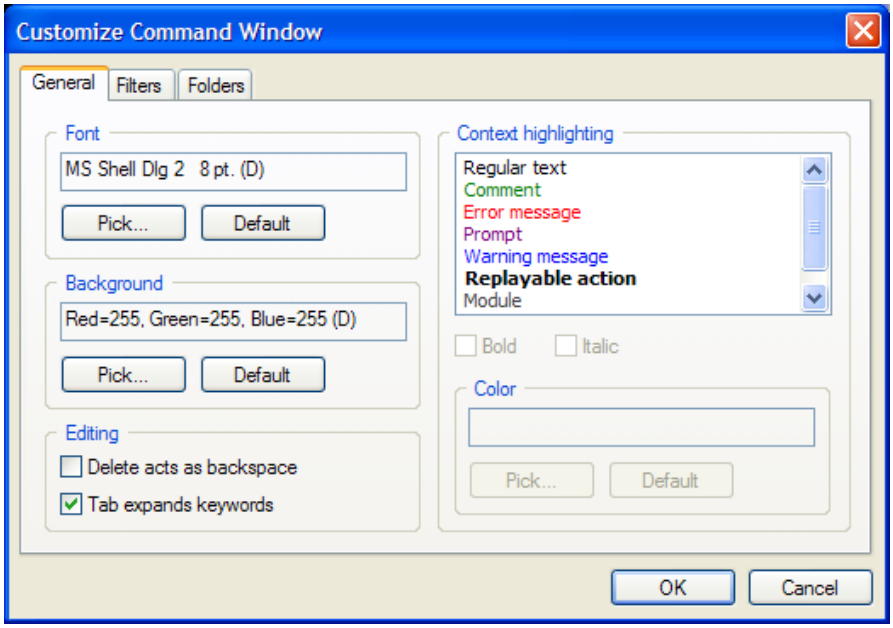
If no path is set in this dialog, then logs are stored in the location set by the TANNERLOGPATH environment variable.

If there is no TANNERLOGPATH environment variable, logs are stored in a folder "Tanner Logs" under the temp folder, which can be %TMP% or %TEMP% or even the WINDOWS folder on your hard drive, depending on the configuration of your Windows environment.

Setting Command Window Text Styles

Right-click in the Command window and select **Customize** (or use the shortcut **F8** while the Command window is active) to open the **Customize Command Window** dialog, then use the **General** tab to set

Command window display characteristics for each of the categories of text—error messages, warnings, modules, etc.



Font Use this field to **pick** a font or set it to the **default**.

Background Use this field to **pick** a background color or set it to the **default**.

Editing Use this field to set the behavior of the backspace and tab keys.

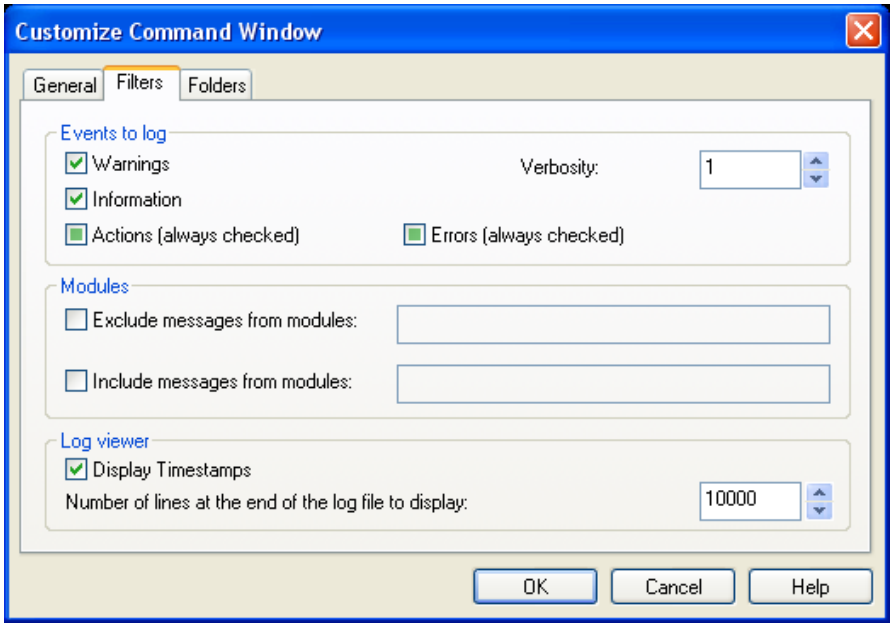
Delete acts as backspace When this checkbox is enabled, the **Delete** key functions like the **Backspace** key by removing text to the left of the cursor. When this checkbox is not checked, the **Delete** key removes text to the right of the cursor as usual.

Tab expands keywords When this box is checked, you can use the **Tab** key to expand all keywords.

Context highlighting When you select a text type in the upper pane, you can check the boxes to make it **Bold** and **Italic**, and also **Pick** a text color or set it to the **Default**.

Filtering Information in the Log File

Use **Customize Command Window > Filters** to limit the type or amount of information written to the log file. You can also set certain display controls for the Command window which do not effect the log file.



Events to log

Check the boxes to include events categorized as **Warnings** and **Information** in the log file and Command window display. (**Actions** and **Errors** are always written to the log file.)

The higher the **Verbosity** value, the higher the detail included in event messages written to the log file.

Modules

Each program module in S-Edit generates a set of messages, identified by a preceding # *module_name*, where *module_name* is a three character abbreviation such as “LIC” for license related messages. Enter these abbreviations, separated by commas, to exclude or include a type of message from the log file and Command window.

When **Exclude messages from module** is checked, messages from any module abbreviations listed in the corresponding entry field will not be written to the log file.

When **Include messages from module** is checked, messages from any module abbreviations listed in the corresponding entry field will be written to the log file.

Log viewer

Note: These options effect the Command window display only, not what is written to the log file.

Check the **Display Timestamps** box to display the timestamp when one is included in a message.

Enter an integer between 100 and 100,000 to set the **Number of lines at the end of the log file to display**.

Arranging Interface Elements

The S-Edit interface is highly flexible, allowing you to display and arrange elements as you choose.

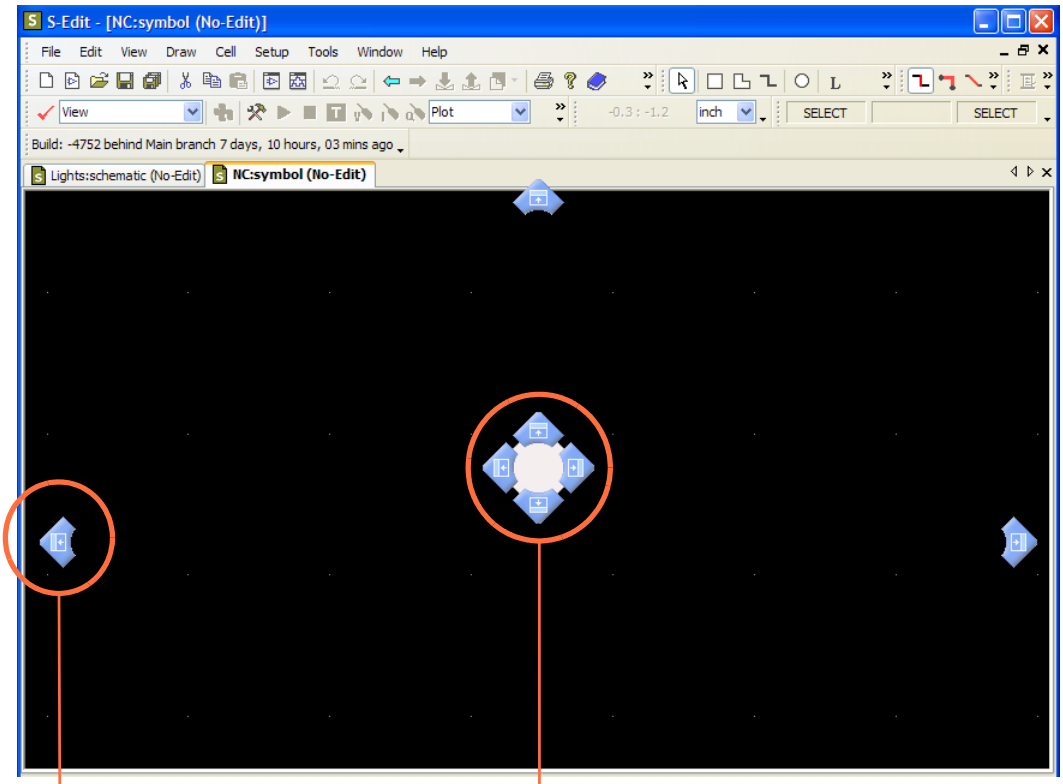
You can dock individual S-Edit toolbars, navigators and the design window, or they can float anywhere in the application window. With the exception of the locator toolbar, toolbar buttons will be arranged vertically when docked to the sides. Design windows can be shown, hidden or made dockable directly from the interface.

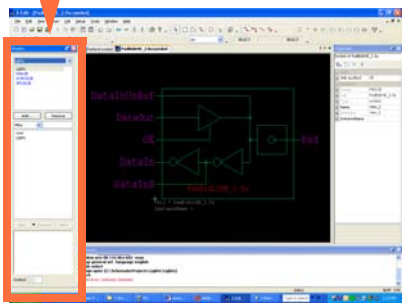
Display colors can be customized for objects, background, the grid, etc., and can be set differently in the text editor for each of the file types it reads. Windows reuse in the design area is configurable, as are parameters such as selection range and behavior.

Settings for these and other configuration parameters are available in the **Setup** menu.

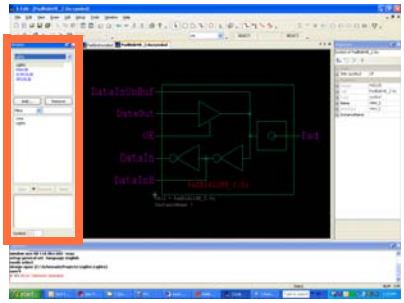
How to Dock Windows in S-Edit

S-Edit uses standard Windows 2000/XP docking behavior. When you drag any of the interface windows and release the cursor over a blue docking arrow, it will dock as shown below.


S-Edit - [NC:symbol (No-Edit)]
File Edit View Draw Cell Setup Tools Window Help
Build: -4752 behind Main branch 7 days, 10 hours, 03 mins ago
Lights:schematic (No-Edit)
NC:symbol (No-Edit)



Edge Arrows
When you release a window over a docking arrow at an edge of the screen, it will dock at the corresponding edge, filling the entire application area.



Center Arrows
When you release a window over a docking arrow at the center of the screen, it will dock on the corresponding side, between the other docked windows.

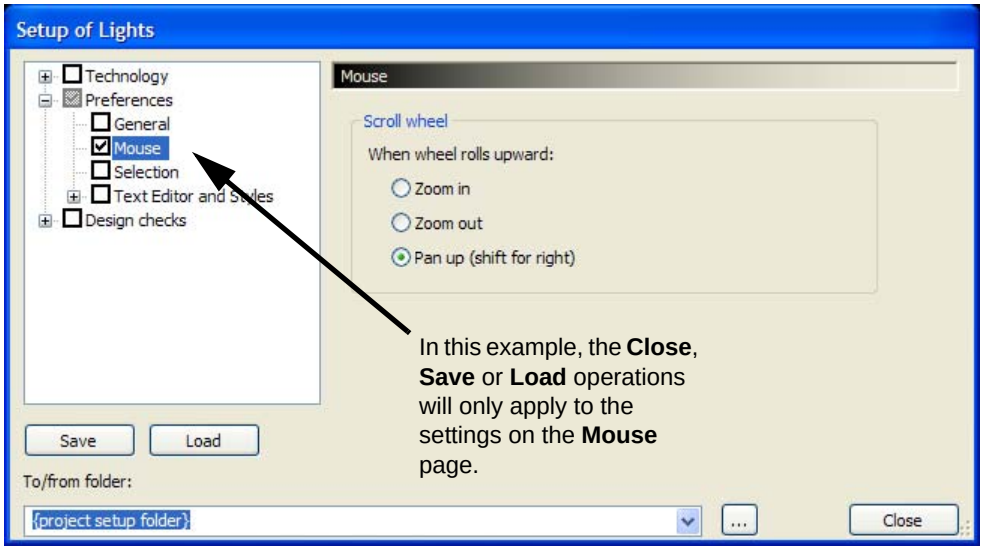
Setup Options

The **Setup** dialog contains configuration controls as follows. Setup values are stored as TCL scripts in a special directory, with one TCL file for each page of the **Setup** dialog (**colors.tcl**, **grid.tcl**, etc.).

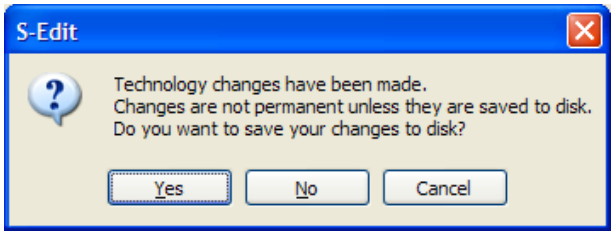
- **Text Editor and Styles** (Text File Update Options, page 47) – Controls text file update and display characteristics.
- **Design Checks** – Controls how individual design checks, pin and port connections are flagged. and which name validation scripts are used. See “Checking a Design for Errors” on page 135.
- **SPICE Simulation** – Controls simulation parameters and options. See “Running Simulations” on page 232.

Saving and Loading Setup Options

If you want setup values to apply *only* to the current editing session, use the **Close** button. If you want setup values to be saved so they can be loaded and reused, use the **Save** button. Note that changes to a setup page will only be applied if the corresponding checkbox is checked.



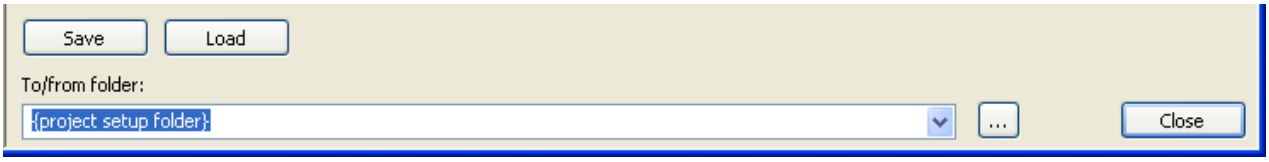
If you have made setup changes without saving them, S-Edit will prompt you to confirm that choice.


S-Edit
Technology changes have been made.
Changes are not permanent unless they are saved to disk.
Do you want to save your changes to disk?
Yes
No
Cancel

You can name and save as many setup configurations as you like. This is useful, for example, so that setup definitions can be copied to a new design file, when you want to save setup values that differ from those S-Edit loads automatically, or so that different design types can have their own setup schemes.

Scripts corresponding to setup dialog settings have special names; putting other scripts in those folders will have no effect. If a script of the same name is present in both locations, settings from the user preferences folder take precedence over settings from the design project folder.

Note: Remember, only settings with their checkbox checked will be saved or loaded.



Save

Place a check in the checkbox for each setup page you want to save and click the **Save** button. S-Edit will save the TCL files to the folder you specify.

Note: Only dialog pages that are checked will be saved.

Load

Place a check in the checkbox for each setup page you want to load and click the **Load** button. S-Edit will load the TCL files from the folder you specify.

Note: Only dialog pages that are checked will be loaded.

To/from folder

Use this drop-down list to select the folder to which a setup file will be saved, or from which a setup file will be retrieved. You can also browse to a directory of your choice to save or load a setup file. Setup scripts are not required to exist in either location, in which case the S-Edit default values are used.

Note: When choosing a folder, it is important to understand the order in which S-Edit will search for and load files; see "Load Order for TCL Files" on page 205 for details.

{project setup folder} is the setup folder in the design folder of the specific design or library shown in the title of the **Setup** dialog. This is the default setup folder for the design.

{user preferences folder} is a setup folder in the Tanner directory on a local computer from which S-Edit will automatically read user-defined setup values. Settings from the **user preferences folder** overwrite the default settings for a design.

Project Setup Folder Location

The predefined location of the project setup folder is **..\<*project location*>\<*designname*>\setup\preferences**, where *project location* is the path and directory where the design is saved.

User Preferences Folder Location

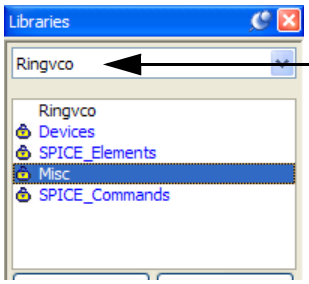
The predefined location of the user preferences folder is **C:/Documents and Settings/<username>/Application Data/Tanner EDA/scripts/open.design/setup**, where *username* is the login name of the current user.

Setup Values by Design versus Library

S-Edit uses the setup of the primary design (which is the design active in the combo box at the top of the Libraries navigator) when editing and checking the design, even on elements instanced from libraries.

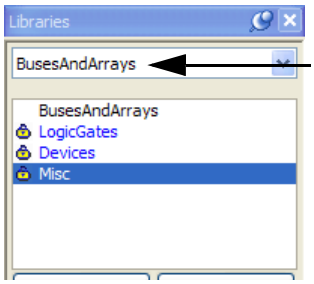
Warning: In earlier versions of S-Edit, if the active view was a library cell, the setup dialog edited the library's setup. Now it edits the setup of the primary design.

If more than one open design uses a given library, the library still uses the setup from whichever design is currently active.



Ringvco is the primary design, **Misc** is one of the its libraries.

In this case, cell views from **Misc** use the **Ringvco** setup.

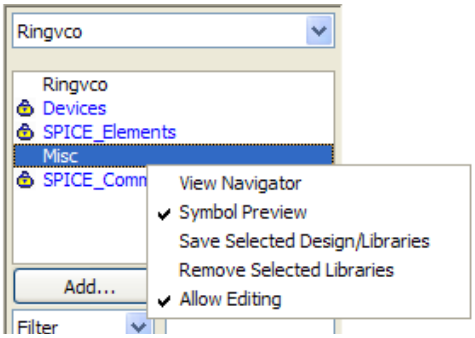


BusesAndArrays is now the primary design. It also uses the **Misc** library.

in this case, cell views from **Misc** use the **BusesAndArrays** setup.

When a library is open, either as the primary design or as a referenced library, it is locked—no other users can *save* changes to that library.

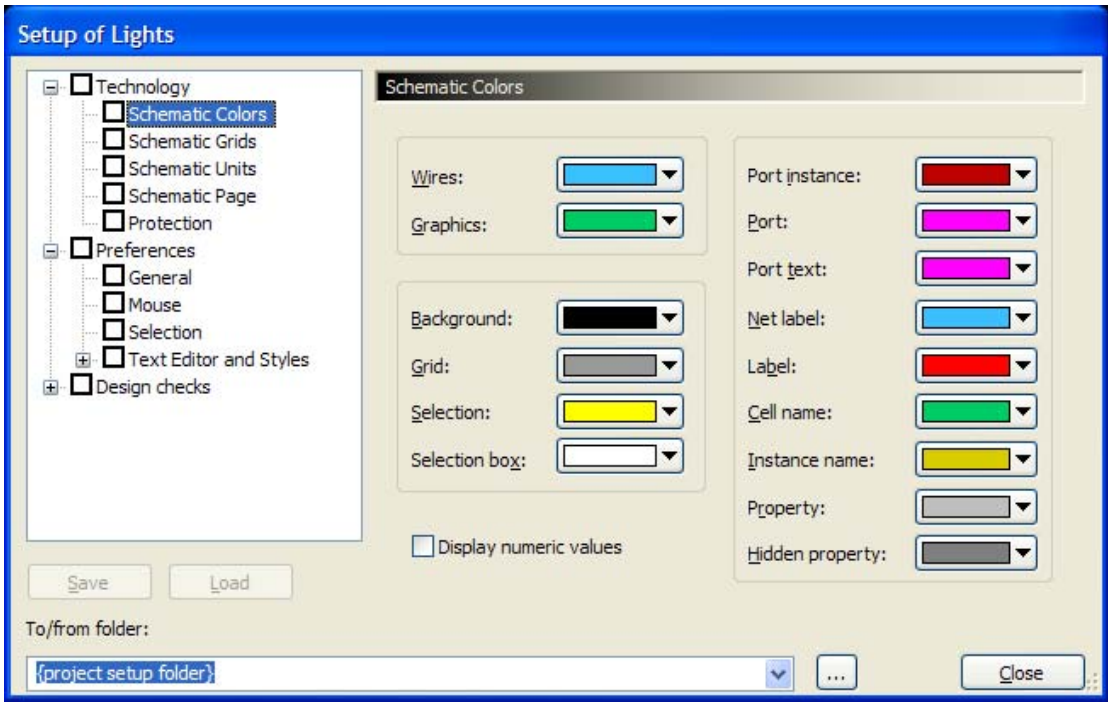
However, other users can *edit* that library as long as the **Setup > Technology > Protection** option **Allow Editing** is enabled. A lock icon next to the library name indicates that edits are not allowed. If you right-click in the Libraries list you will open a context menu where you can toggle the **Allow Editing** setting on and off. The lock icon will display accordingly.



Note that the **Allow Editing** setting, when initiated from the Library navigator, is not a permanent setting – it is a temporary setting that only applies while the current file is open.

Setting Display Colors

Setup > Technology > Schematic Colors controls the color of objects in the design window



Schematic Colors

Use these fields to set the color for the associated drawn element (wire, port, background, grid, etc.).

When you click on the drop-down arrow S-Edit opens the standard windows color dialog, shown below:

Display numeric values

Check this box to also show the six-digit hexadecimal representation of colors in the form RRGGBB, corresponding to the red, green, and blue values of the color.

Grid and Unit Coordinates in S-Edit

S-Edit uses *display units* to report object dimensions and coordinates, to set a grid to use as a visual aid while drawing, and to establish an optional *snapping grid* for the mouse cursor.

Major and Minor Display Grid

The grid:display grid consists of two arrays in the design area, large (*major*) and small (*minor*), that can be set to any desired size to provide a drawing guide, typically to set minimum feature size or critical manufacturing measurements. The grids can be independently displayed or not, as either dots or lines.

Display units can be shown in millimeters, centimeters, meters, or inches. The choice of display units does not affect the scaling of your design. If you change the display units, for example, from millimeters to inches, S-Edit will automatically convert the unit values displayed in the locator bar to inches, but nothing in the design itself will change.

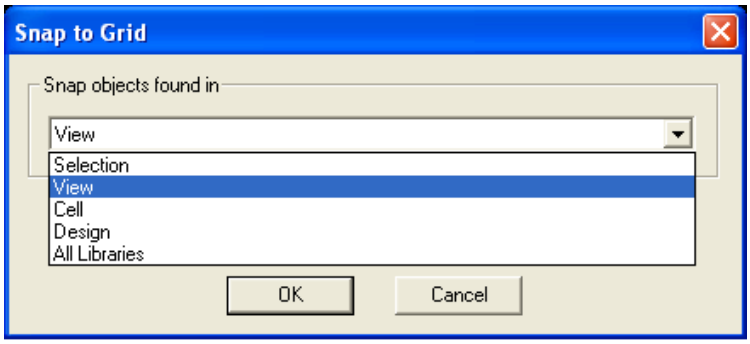
The apparent spacing of the grids will vary with the magnification of the work area. If the number of screen pixels per grid square falls below the value entered in this field, the grid is hidden.

The coordinate origin point (0,0) is indicated by a large cross-hair marker, and its display can also be toggled on and off.

Snap Grid

The snapping grid, which can be a different size than the display grid, causes all drawing and editing coordinates entered with the mouse to be placed on grid points. To achieve adequate resolution, you may wish to adjust the spacing of the mouse snap grid based on your minimum feature size.

Sometimes when the grid setting has been changed or when a symbol is flipped or mirrored, symbols may be shifted off grid. If so, you can use the **Draw > Snap to Grid** command to snap them back to grid.



The snapping to grid action can be applied to the standard Tanner scope of selection, from **View** to **Cell** to **Design** to **All Libraries**.

Internal Units

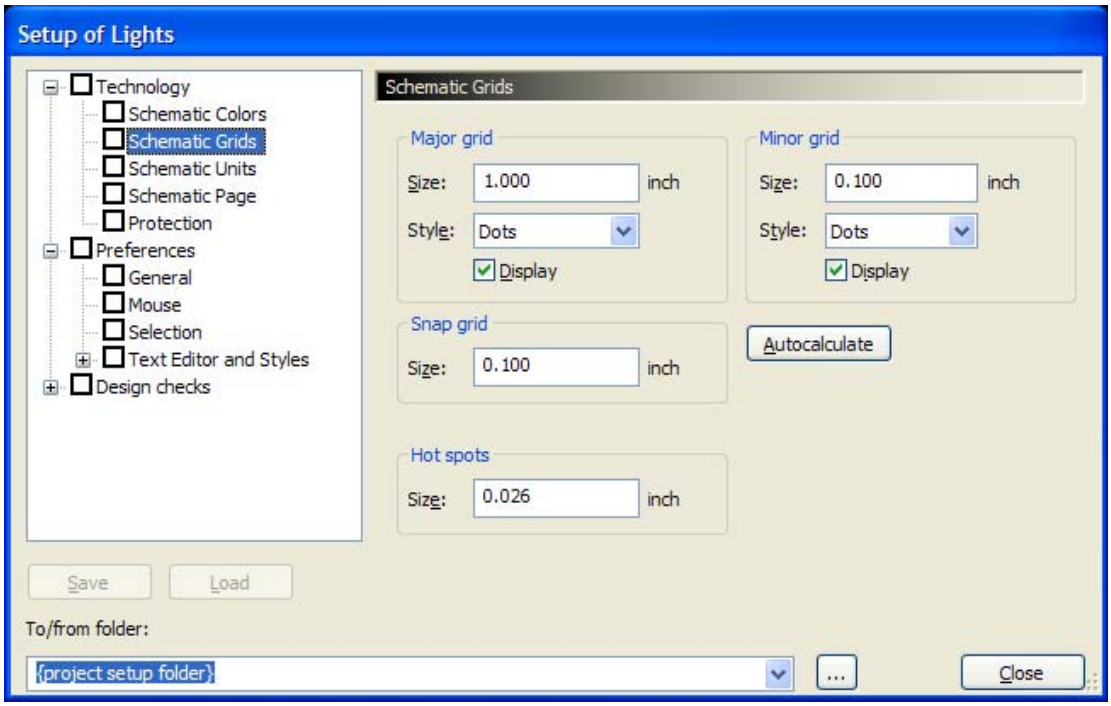
For its own computation, S-Edit uses *internal units*. Before beginning your design, you must define the relation between internal units and *physical* units. This ratio will determine the maximum dimensions of the design area and the smallest object that can be drawn. This relation is also critical when you replace your design setup or export a design, since it sets the scale of the design file.

The S-Edit design area extends from -536,870,912 to +536,870,912 internal units in the *x*- (horizontal) and *y*- (vertical) directions. Thus, if 1 internal unit = 0.001 millimeters, the largest possible design would be 1,073,741 internal units (roughly 42.3 inches) on a side. Similarly, the smallest dimension S-Edit can define is 1 internal unit. If 1 internal unit = 0.001 millimeters, the smallest possible feature size would be 0.001 millimeters.

To set internal units see “Setting the Size of the Display and Snap Grids,” below.

Setting the Size of the Display and Snap Grids

Setup > Technology > Schematic Grids controls the size and style of the various grids.



Major Grid (in display units)
The absolute spacing of the major grid display. The value entered in this field is the distance, in display units, between major grid points.

Minor Grid (in display units)
The absolute spacing of the minor grid display. The value entered in this field is the distance, in display units, between minor grid points.

Snap Grid
The absolute spacing of the cursor snap grid, entered in display units as the length of one side of a grid square.

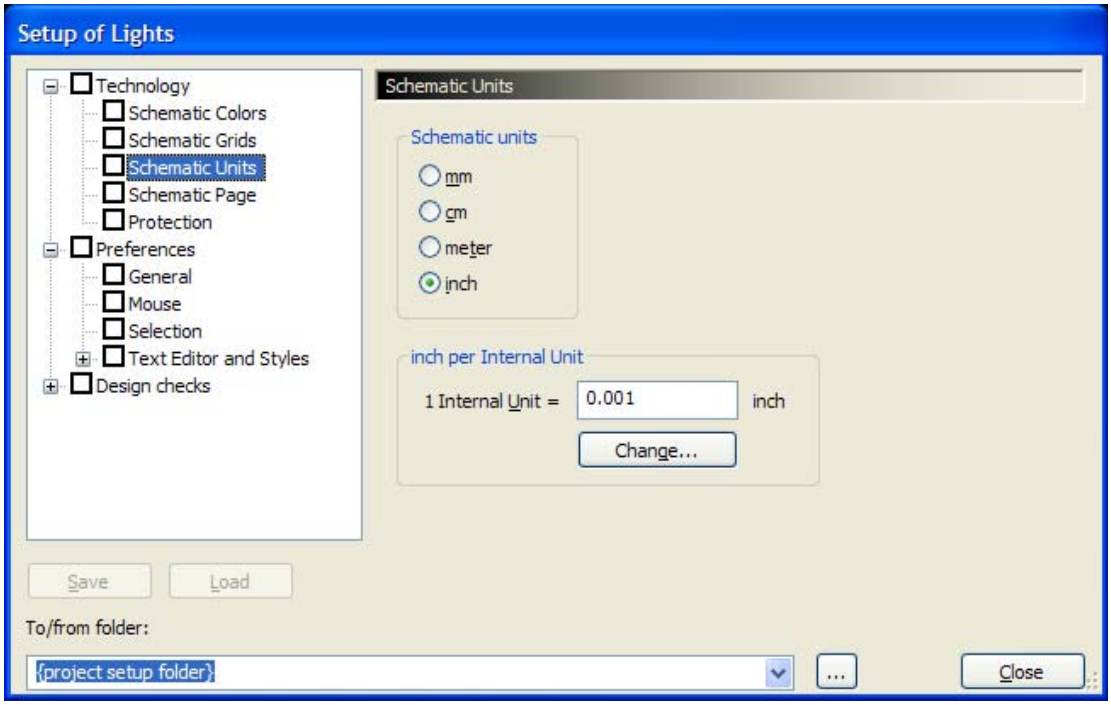
The value entered in this field is the minimum resolution, in display units, allowed during drawing and editing operations. All drawing and editing coordinates are snapped to this grid size when it is not zero.

Autocalculate
Pressing this button calculates the largest possible minor grid size for the selected design based on all port positions in the design and updates the other values in the dialog. If a common grid value cannot be found S-Edit will return one (1) internal unit.

Hot spots
Use this field to set the hot spot size for ports, in display units. (See “Hot Spots” on page 121.) The default value is eight internal units when a new design is created. Zero is an allowed value.

Setting Unit Type and Scaling Display Units

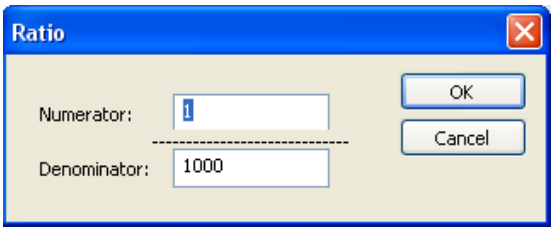
Use **Setup > Technology > Schematic Units** to establish the relation between S-Edit display units and physical units.



Schematic Units Choose a physical unit for your design.

***units* per Internal Unit** Enter the scale of physical units to internal units as a decimal value.

Click **Change** to enter the ratio as a fraction, then enter a numerator and denominator value.

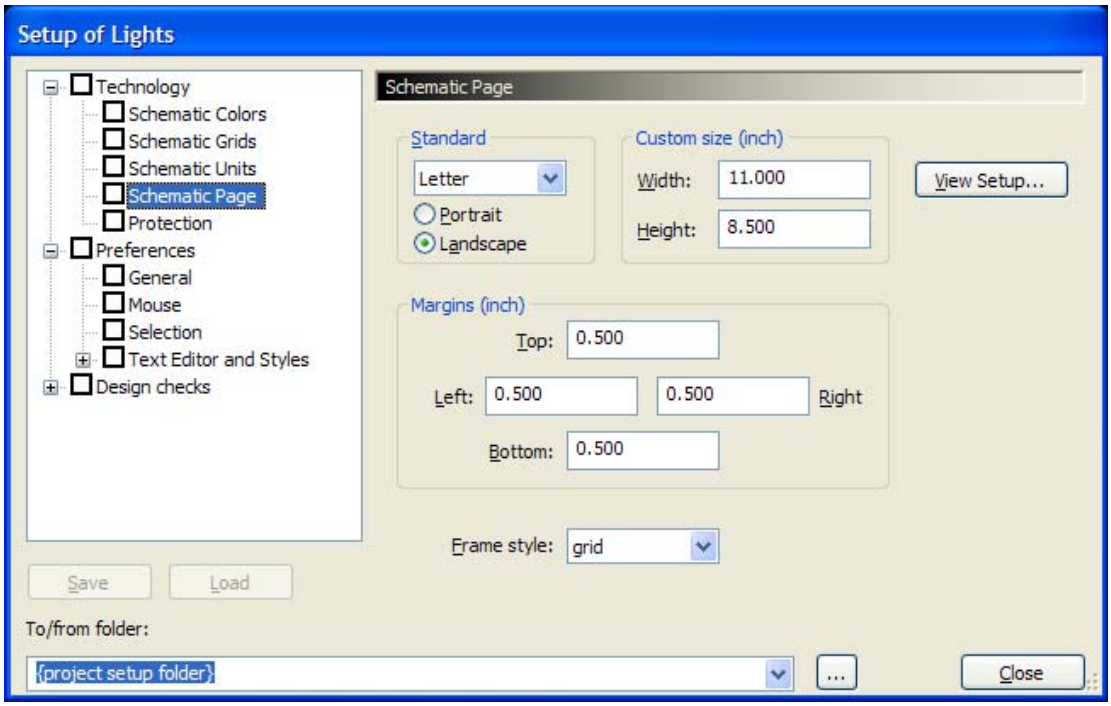


Scaling Solder Dots

Solder dot size is a function of the internal unit-to-physical unit scaling. By default, solder dots are 1 mm wide, buses are 0.7 mm wide and solder dots on buses are 1.6mm. However, if you are having trouble seeing your solder dots, you can use the tcl variable **tanner_bus_scale** to adjust the built-in solder dot algorithm. The default value = is 1.0; a value great than 1 will produce larger solder dots, buses, and bus solder dots (for example, set tanner_bus_scale 1.5).

Setting Page Size

Setup > Technology > Schematic Page sets size of each page, the margins and border styles.



Standard — Choose from the predefined page sizes, or select **Custom** to define your own.

Custom size — Enter the **Width** and **Height** here when using a Custom page size.

Margins — Controls the margins between the frame and paper edges, separately for **Top**, **Bottom**, **Left** and **Right.**

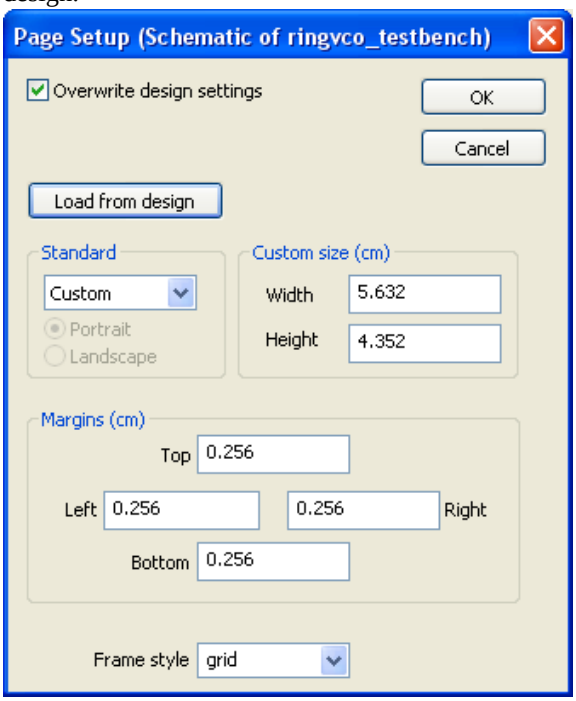
Frame style — Choose from the options **box**, **grid** or **none**. Grids increment as an alpha array in the x direction from A and numerically in the -y direction from 1.

All frames are placed with the lower left corner at the origin.

S-Edit calculates the frame size with respect to the unit settings defined in **Setup > Technology > Schematic Units** so that when you print using the **Do not scale** setting, the design window will print to fit the page size you have chosen.

View Setup

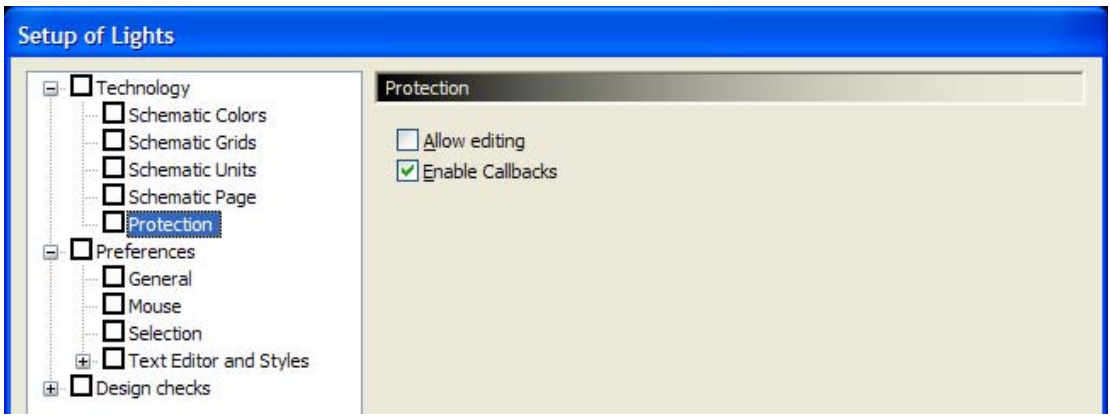
Opens a dialog that allows you to enter page setup values for the currently active view that differ from those defined for the entire design.



- Check **Overwrite design settings** to apply the values in this dialog to the active view.
- Click on **Load from design** to reset these page values to those defined for the entire design.

Setting Cell Protection and Enabling Callbacks

Setup > Technology > Protection allows you to prevent editing and enable callbacks for an entire design file.



Allow editing

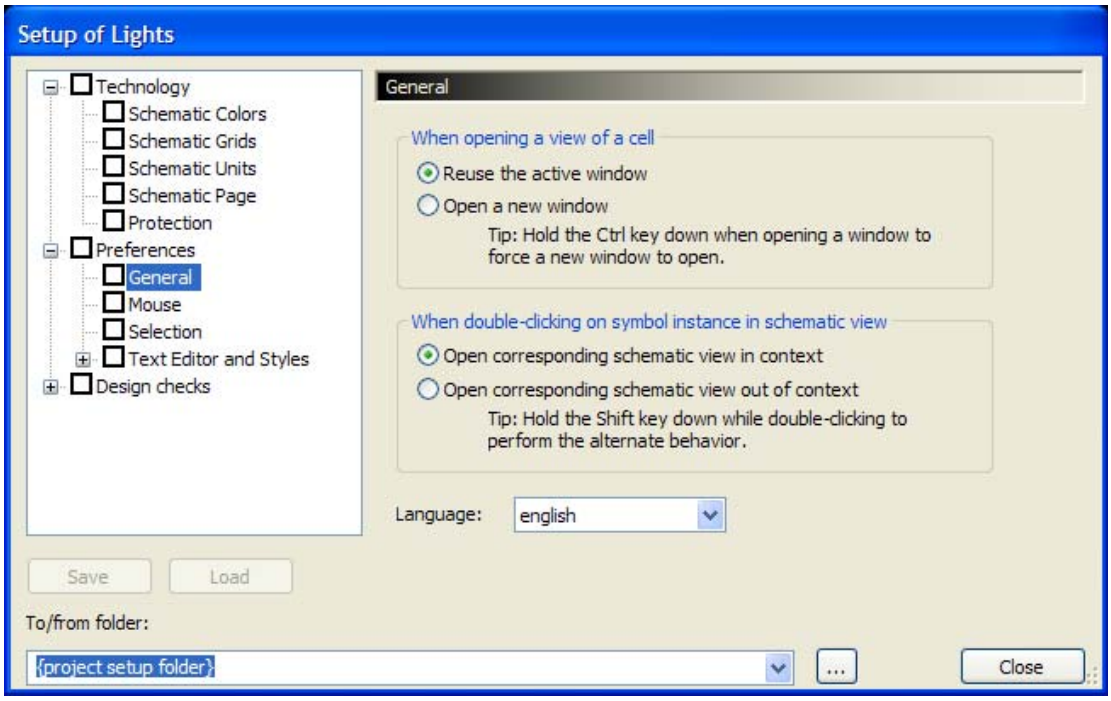
When checked, cell contents in the design file can be edited.

Enable Callbacks

When checked, allows callback operation for the entire design file. See “Callbacks” for Property Values, page 155.

Window Reuse and Language Selection

Use **Setup > Preferences > General** to set design window reuse and language display.



When opening a view of a cell

Reuse the active window sets S-Edit to replace the contents of the active design window when opening a new cell. If you do not select a view type, the new cell will clone the current view type.

Open a new window sets S-Edit to open a new design window each time you open a new cell view.

Note: Holding the **Ctrl** key down while opening a cell will force it to open in a new window.

When double-clicking on symbol instance in schematic

Use this option to set whether a double-click in the design window opens a unique instance or the primitive cell symbol.

Use **open corresponding schematic view in context** to push down to a specific instance when you double-click on a symbol.

Use **open corresponding schematic view out of context** to show the primitive symbol when you double-click on a symbol.

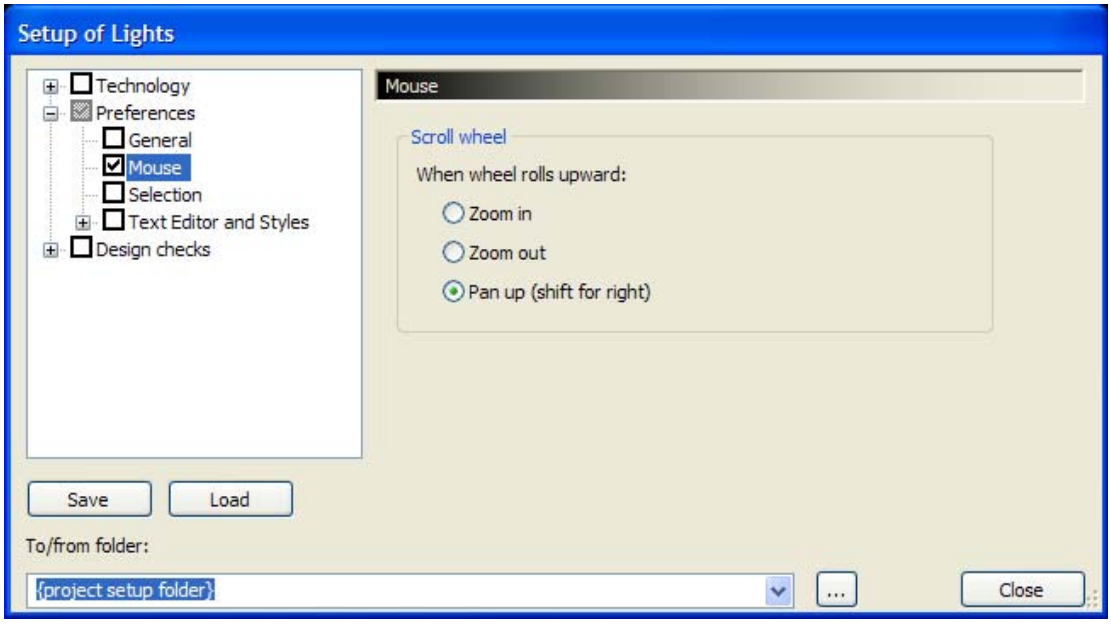
Note: Holding the **Shift** key while you double-click toggles the action to the opposite of the current setting.

Language

Select a language for menu and dialog text from the drop-down list. Options are American English, Japanese and Simplified Chinese.

Mouse Wheel Options

Use **Setup > Preferences > Mouse** to set how the design window view changes when you spin the mouse wheel away from your hand.

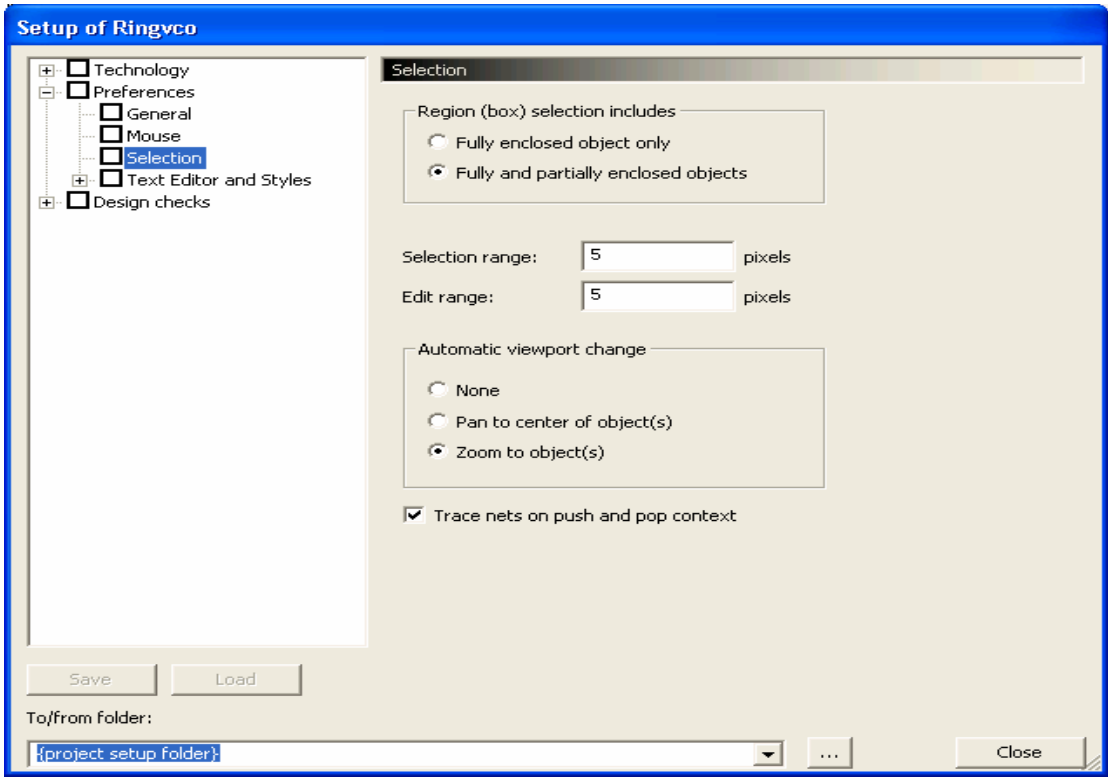


Scroll wheel

- **Zoom in**—zooms in to the active view to magnify the view by a factor of three.
- **Zoom out**—zooms out of the active view by a factor of three.
- **Pan up (shift for right)**—pans the design up with an upward spin of the mouse wheel or down with a downward spin. Hold the **Shift** key to pan right with an upwards spin or left with a downwards spin.

Selection Behavior Options

Setup > Preferences > Selection governs mouse selection behavior in the design area and from hyperlinks in the Command window.



Region (box) selection includes

Fully enclosed object only—with this option enabled, only objects completely contained within a selection box will be selected.

Fully and partially enclosed objects—with this option enabled, all objects completely or partially enclosed by a selection box will be selected.

"Selection Behavior Options" on page 46

A positive integer *x* such that when a mouse button is clicked less than *x* pixels from an object, though not touching it, the object will still be selected.

Edit Range

A positive integer *e* such that if the pointer is within *e* pixels of an edge or vertex of a selected object, the default operation of the MOVE-EDIT button is an **edit**. Outside this range it is a **move**.

Automatic viewport change

Controls the display behavior in the design area when you execute the Find command. **Zoom to object(s)** is the default.

- **None**—simply selects objects.
- **Pan to center of object(s)**—If the selected objects fit in the current view entirely, pans to center on them without changing zoom level. If the objects do not fit on screen at the existing zoom level, uses the "zoom to objects" algorithm described below.
- **Zoom to object(s)**—pans and zooms so the MBB of the selected objects occupies approximately 1/3 of the screen, vertically. This setting will not zoom in any closer than a "home" view.

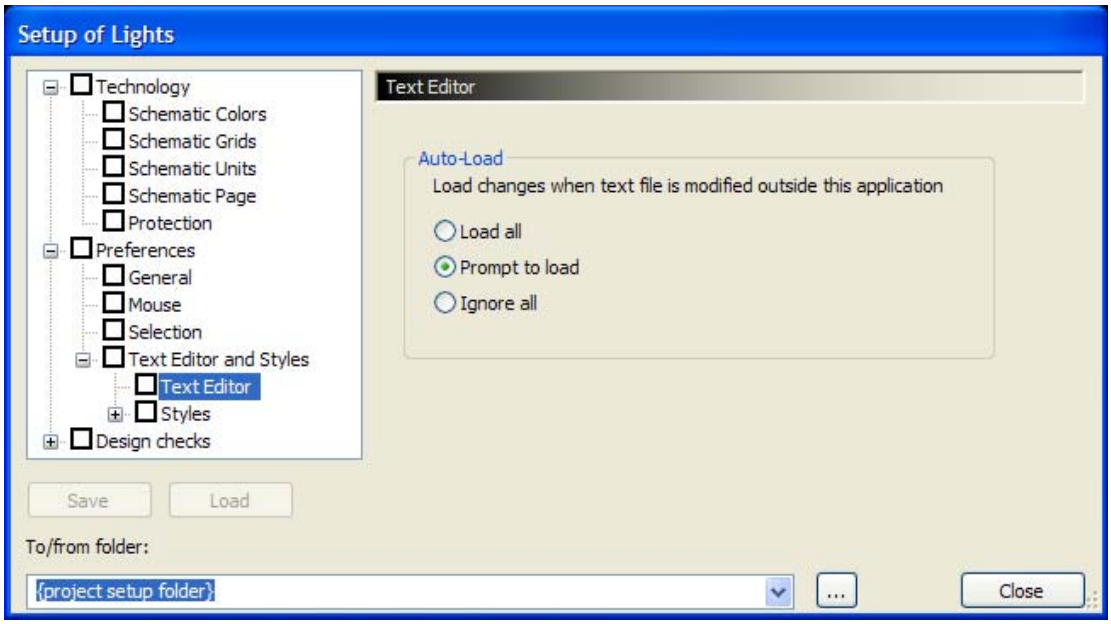
Trace nets on push and pop context

When checked, net highlighting is maintained when you push into and pop out of views. Disabling this feature speeds push/pop performance on larger designs.

Text File Update Options

The **Setup > Preferences > Text Editor and Styles > Text Editor** dialog governs if and how open text files are updated when they are saved outside the application.

The S-Edit text editor checks the stored version of a file for modifications whenever files are saved, first changed, and when the text window or application becomes active or is closed. If a file has not been modified outside S-Edit, nothing will happen. If a file has been modified outside S-Edit, the selected action is triggered.



Auto-Load

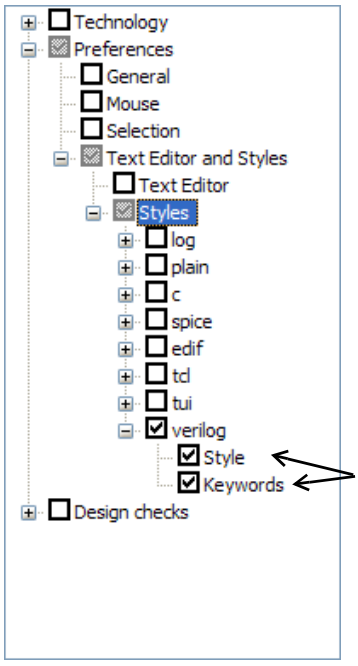
Load all—S-Edit automatically updates text files that have been modified outside of the text editor.

Prompt to Load—(default) If a text file is modified outside S-Edit, you will be prompted to accept or ignore those changes.

Ignore all—externally modified files are not updated to S-Edit.

Text File Display Options

Setup > Preferences > Text Editor and Styles > Styles sets the text display characteristics for different file formats.



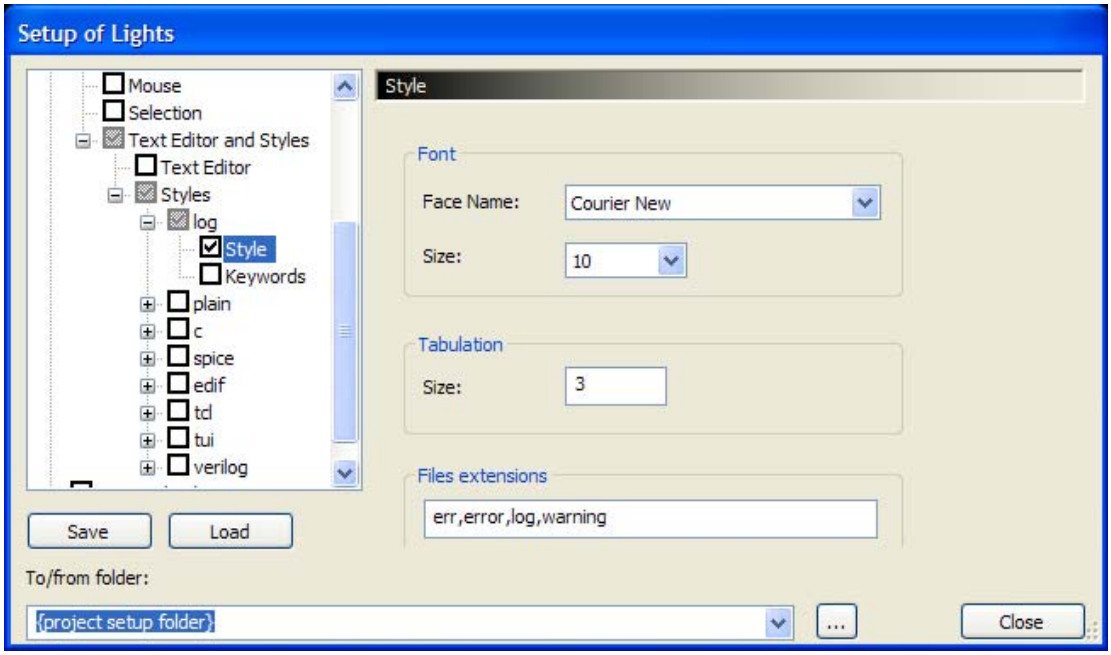

```
module single_pin( in );
  input in;
endmodule

module test_verilog( in, out, out2, out3 );
  input in;
  output out, out2, out3;
  assign out2 = 1'b0;
  assign out3 = 1'b1;
  nand2 X1( .A(1'b0), .B(in), .Y(out) );
  nand2 X2( .A(1'b1), .B(in), .Y(out) );
  single_pin X3( .in(in) );
endmodule
```

There are two control pages, **Style** and **Keywords**, for each file type that can be set for special format display in the text editor.

For example, in the simple Verilog file shown above, keywords are formatted blue.

Setup > Preferences > Text Editor and Styles > Styles > *{filetype}* **> Style** sets the default font attributes for each predefined file type.



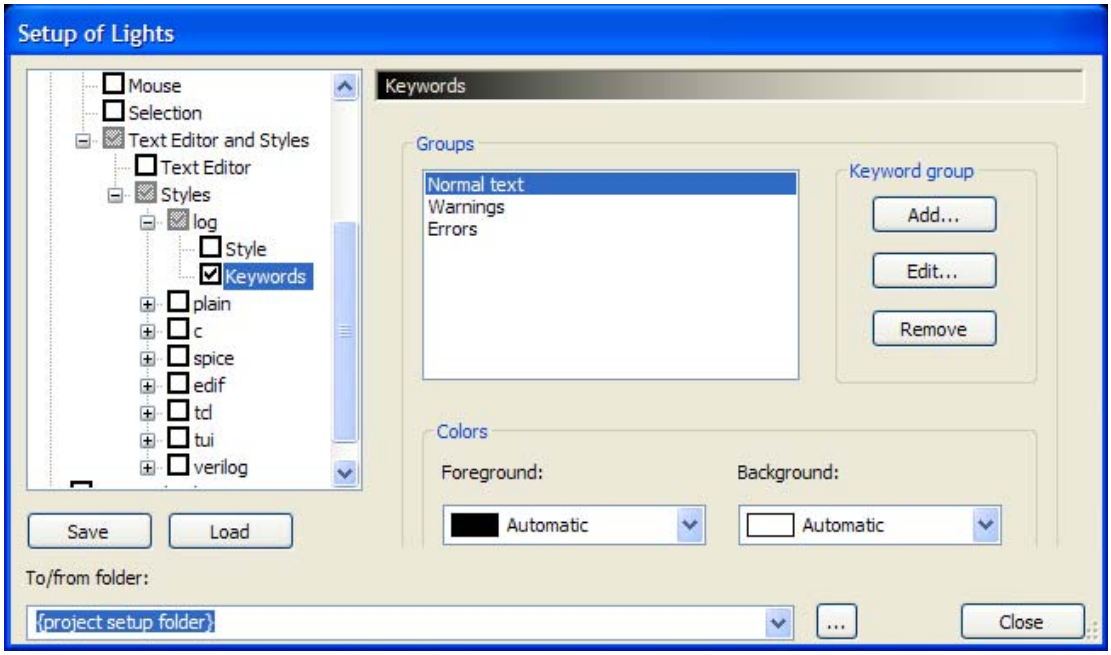
Font — Select or enter a font from the **Face Name** drop-down menu and a point size in the **Size** field.

Tabulation — Enter a positive integer value to set the increment, in spaces, of the tab spacing the text editor uses.

File extensions — Enter the extensions of the files that S-Edit should include in the active category, separated by commas with no spaces.

Setup > Preferences > Text Editor and Styles > Styles > *{filetype}* **> Keywords** sets how different categories of information, called *keyword groups*, are displayed.

Keyword groups are categories of text for which you can set display characteristics in the S-Edit text editor. For example, you can define the keyword group “warnings” for EDIF files, and set S-Edit to display it in red text against a yellow background. Each file type has a set of predefined keyword groups that cannot be edited or deleted.



Groups Displays the keyword groups defined for a given file type.

Keyword group Use **Add** to enter the name of a new keyword group. Use **Edit** to enter the terms belonging to a keyword group. Use **Remove** to delete a keyword group.

Colors Use **Foreground** and **Background** to set the respective colors for a keyword group.

Design Check Options

Please refer to “Checking a Design for Errors” on page 135.

SPICE Simulation Options

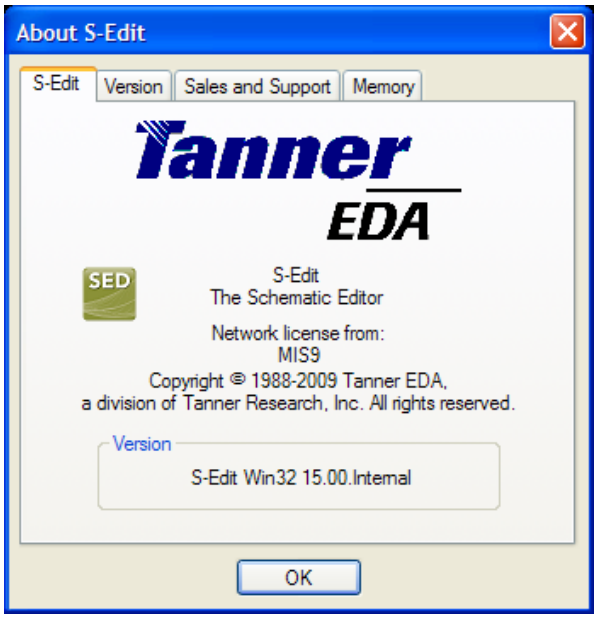
Please refer to “SPICE Simulation Settings” on page 208.

S-Edit Documentation

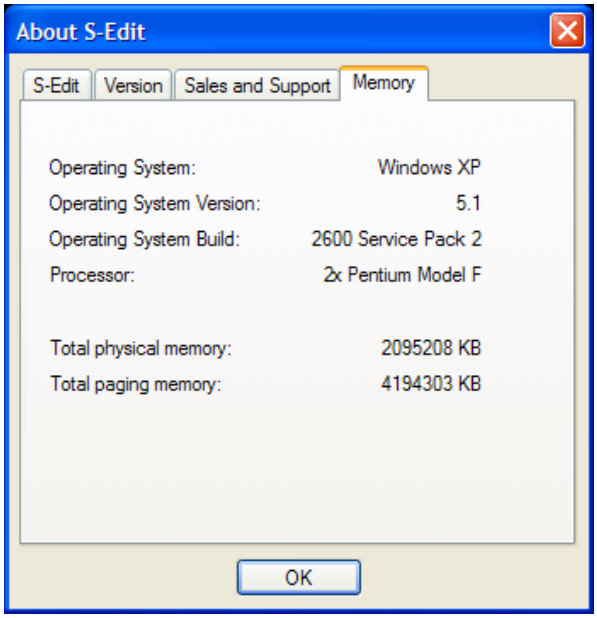
In addition to this manual in PDF format, S-Edit is shipped with application notes, release notes and a tutorial that highlights basic schematic entry and editing operations.

S-Edit Product Support

Use **Help > About S-Edit > S-Edit** to view the product version you are using.

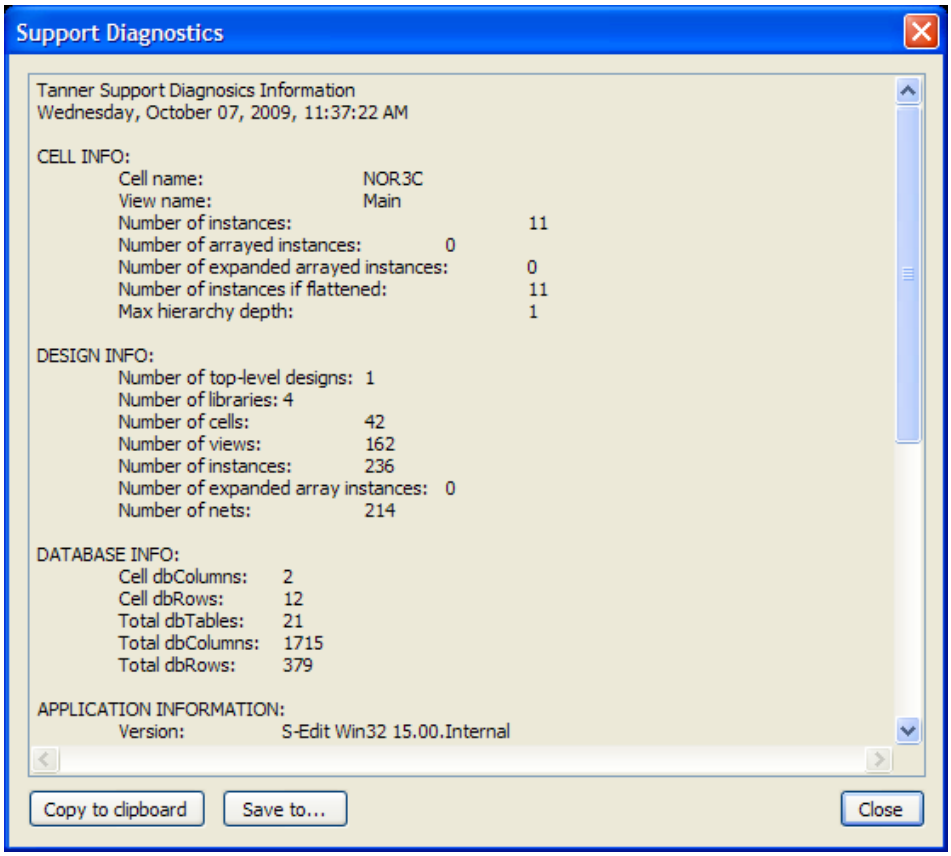


Use **Help > About S-Edit > Memory** to display the operating system, processor and memory information for the computer you are using.



Support Diagnostics

Tanner EDA customer support they will sometime request the comprehensive design, system and hardware information found in **Help > Support > Support Diagnostics**.



2 Creating a Project

The highest level entity in the S-Edit schematic database hierarchy is the design. A design contains many cells, some of which may be referenced from a library.

Most often a cell will contain a single *interface*, which contains a single symbol view and a single schematic view. However, a cell can contain any number of interfaces, and each interface can contain any number of symbol views and schematic views.

Project Structure in S-Edit

It is important to understand the basic project structure and terminology used in S-Edit.

Design

A *design* is the container for all elements of the design database.

Library

A *Library* is also a design, one whose cells are externally referenced by other designs. A design can reference multiple libraries, and any given library can be referenced by multiple designs.

Cell

A *cell* is the fundamental unit of design. A design contains multiple cells. Cells in turn contain multiple *views*, of different types. Cells can be instanced by other cells (an instance is a generic reference to a representation of cellB found within cellA.)

Properties

A property is an attribute of an object (cell, instance, shape). S-Edit differentiates between built-in properties (e.g., cell name, shape type) which are always defined, and user-defined properties (e.g. L = on a transistor cell) which are optional and not interpreted directly by the engine. Each type of object has its own set of parameters, which are displayed and can be edited in the Properties navigator. Please refer to “Symbol Property Types” on page 147 for a complete discussion.

View

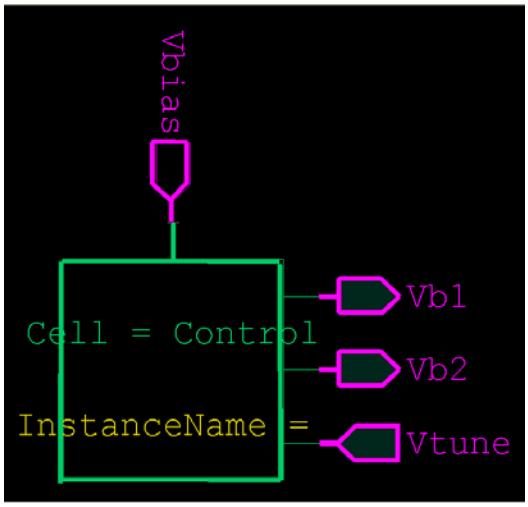
A *view* is simply a component of a cell definition. Each view provides a different way of depicting a cell. S-Edit *view types* are *symbol*, *schematic*, and *interface*. Since T-Spice can run Verilog-A and Verilog-AMS, simulating a schematic with subcells that have either a Verilog-A or AMS description will simulate the Verilog code; so S-Edit has *SPICE*, *Verilog-A* and *Verilog-AMS* views.

Cells can instance each other, but cyclical cell references are not allowed.

View Types

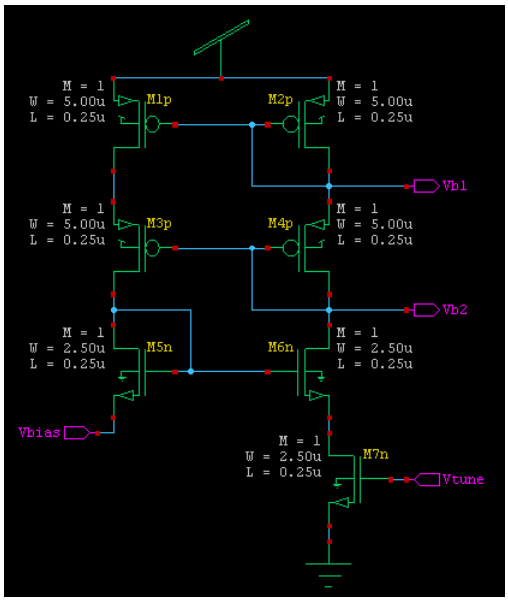
- **Symbol View**

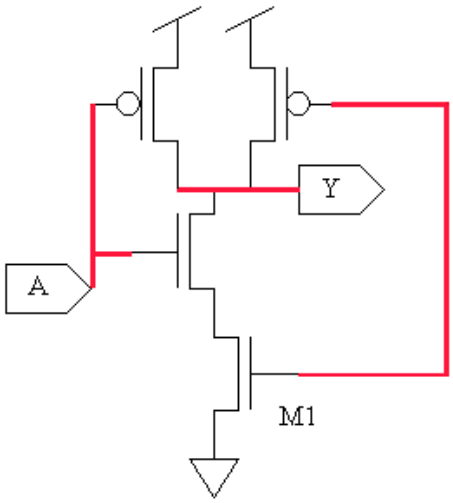
A graphical description of a cell, for use in schematic views of other cells. A symbol view contains the ports of a cell and non-electrical geometry that is representational only. Usually the most basic design components will have only a symbol view.



- **Schematic View**

A more detailed view of a cell, showing ports, instances, references to ports in the parent cell of instanced cells, connecting wires and graphic objects (e.g. boxes, polygons, paths, text labels) that have no electrical meaning.





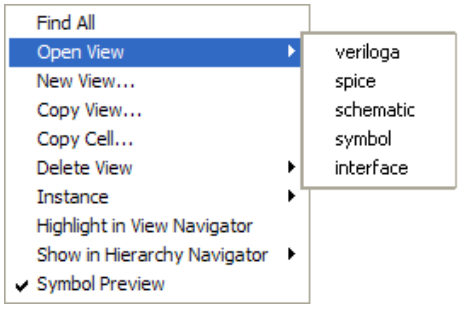
In this example, NAND2 is a two input NAND gate. A, B and Y are ports.

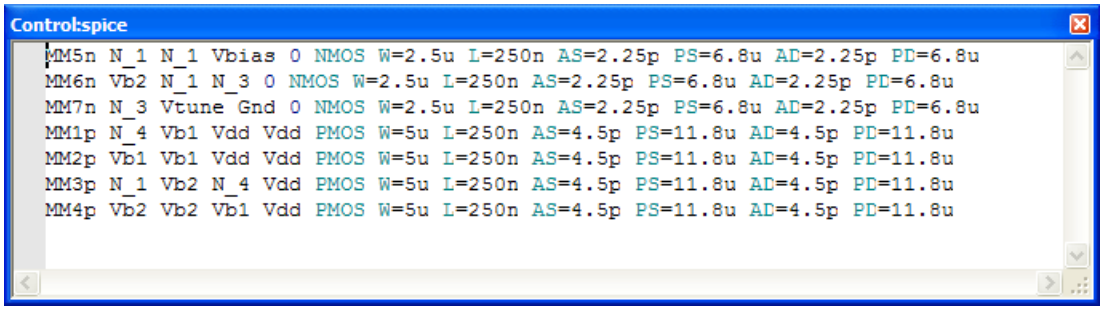
There are seven instanced symbols (including M1, three other transistors, and three power/ground symbols), and six red shapes implementing three nets.

There is a fourth net between the two NMOS devices, and a fifth (power) and sixth (ground) net as well.

- **SPICE and Verilog Views**

You can use **Open View** in the right-click menu in the Libraries navigator to view the SPICE, Verilog-A or Verilog-AMS files associated with a cell in the S-Edit text-editor.



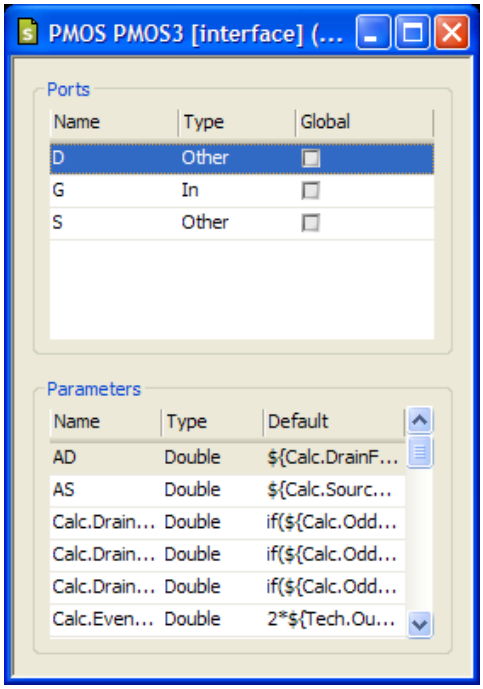

Control:spice
MM5n N_1 N_1 Vbias 0 NMOS W=2.5u L=250n AS=2.25p PS=6.8u AD=2.25p PD=6.8u
MM6n Vb2 N_1 N_3 0 NMOS W=2.5u L=250n AS=2.25p PS=6.8u AD=2.25p PD=6.8u
MM7n N_3 Vtune Gnd 0 NMOS W=2.5u L=250n AS=2.25p PS=6.8u AD=2.25p PD=6.8u
MM1p N_4 Vb1 Vdd Vdd PMOS W=5u L=250n AS=4.5p PS=11.8u AD=4.5p PD=11.8u
MM2p Vb1 Vb1 Vdd Vdd PMOS W=5u L=250n AS=4.5p PS=11.8u AD=4.5p PD=11.8u
MM3p N_1 Vb2 N_4 Vdd PMOS W=5u L=250n AS=4.5p PS=11.8u AD=4.5p PD=11.8u
MM4p Vb2 Vb2 Vb1 Vdd PMOS W=5u L=250n AS=4.5p PS=11.8u AD=4.5p PD=11.8u

The difference between Verilog-A and Verilog-AMS is the file extension when the cell is netlisted. Some simulators, including the S-Edit AMS simulator, handle files differently depending on whether they have .v, .va, or .vams extensions, so we need a way for the user to specify which type of code they will be writing.

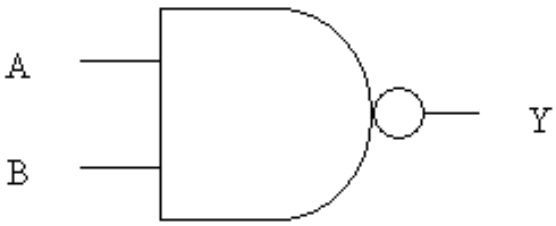
Verilog-AMS views are written as **CellName.vams**, Verilog-A views are written as **CellName.va**. Verilog view types are written as **CellName.v**. You should use Verilog should be used for digital RTL, and structural Verilog (netlists), use Verilog-A for Verilog-A code and use Verilog-AMS for code that mixes Verilog-A and Verilog-D constructs in the same cell.

- **Interface View**

An interface view shows the definition of the electrical interface of a cell, containing ports, optional permutability information for those ports, and a set of user-defined parameters.



Each schematic view and symbol view in a cell must be associated with a specific interface. When you create an instance of a cell you must specify the interface it belongs with. Cells may be associated with multiple interfaces. For example, the NAND2 cell shown below could have two interfaces:



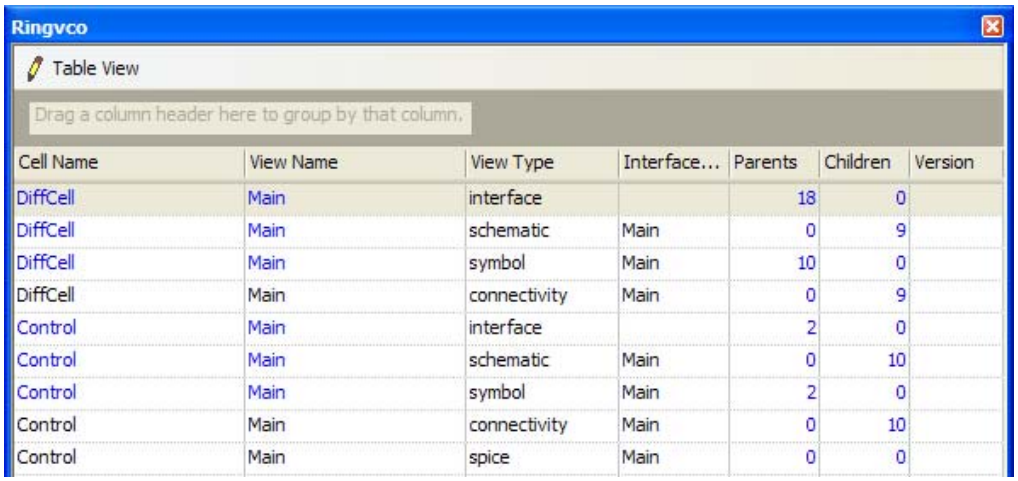
Interface 1
Without power, there are three ports: A = in, B = in, Y = out. A and B are permutable.

Interface 2
With power, there are five ports: A = in, B = in, Y = out, Vdd = inout (virtual) Gnd = inout (virtual). A and B are permutable.

Finding and Prioritizing View Types

S-Edit lets you change the representation of a cell depending on how you want to use it. For example, you may use a Verilog-A view for top-down behavioral design but a schematic view when designing at a transistor level.

If you right-click on a cell name in the Libraries or Hierarchy navigators, you can use the **Highlight in View Navigator** command to see all the view types defined for a given cell.



Cell Name	View Name	View Type	Interface...	Parents	Children	Version
DiffCell	Main	interface		18	0	
DiffCell	Main	schematic	Main	0	9	
DiffCell	Main	symbol	Main	10	0	
DiffCell	Main	connectivity	Main	0	9	
Control	Main	interface		2	0	
Control	Main	schematic	Main	0	10	
Control	Main	symbol	Main	2	0	
Control	Main	connectivity	Main	0	10	
Control	Main	spice	Main	0	0	

Similarly, the right-click menu when you are in the Libraries navigator has a **Show in Hierarchy Navigator** option and the right-click menu when you are in the Hierarchy Navigator has a **Highlight in Library Navigator** option.

How to Use Multiple Views

It is often useful to create a cell with multiple interface, symbol, or schematic views. The following scenarios provide examples of such situations.

Scenarios 1 and 2 are functionally the same. In scenario 1, two separate MOSFET symbols are created, each with a different number of pins. MOSFET here is a primitive device which has no schematic; S-Edit allows you to use either of these symbols.

In scenario 2, two separate symbols are created, for a single schematic with 400 ports. One symbol has all the 400 ports and the other symbol has only 100 ports. This is useful if you want to instance the symbol with 100 ports and not have any dangling nodes errors when you run a design check, as the100 port symbol in this case is still referring to the 400 pin schematic. As such, scenario 2 can have two separate interfaces.

Interface views are also used to store netlist information, for example when symbol ports are generated from a SPICE file when a schematic view does not exist.

Scenario 1

In this example, you want to create a symbols for a 3-terminal MOSFET and a 4-terminal MOSFET, for inclusion in a library.

To do so, you would create two different representations, a 3-pin and a 4-pin interface, each having its own symbol, of a single primitive MOSFET which has no schematic. Library users can then instance either symbol in their designs.

Cell name	NMOS
Interface name 1	NMOS3
Symbol name 1	NMOS3

Interface name 2	NMOS4
Symbol name 2	NMOS4

Scenario 2

In this case, the goal is two different representations (400-pin and 100-pin) from a single schematic view. Suppose you have a component called CORE with 400 terminals. You want one symbol that exposes all the terminals, and another that exposes a subset of just 100 of the terminals. You would create a single cell with one interface, one schematic—and two symbols. One symbol has all the ports, the other symbol has the subset of 100 ports.

Cell name	Core
Interface name	Interface
Symbol name 1	Symbol_400
Symbol name 2	Symbol_100
Schematic name	Schematic

Scenario 3

A library designer has a standard cell library with a single implementation of each cell, and wishes to provide two different pictorial representations of each cell for use as symbols, to adhere to the pictorial representations from two different standards organizations. He should create a single interface, a single schematic, and two symbols for each standard cell, both symbols having the same ports.

Cell name	NOR
Interface name	Interface
Symbol name 1	ACM_symbol
Symbol name 2	IEEE_symbol
Schematic name	Schematic

Scenario 4

You have a primitive component, an NMOS transistor, that you wish to establish with two sets of default properties on the symbol. You would create a cell with multiple symbols, each symbol having a different interface for a different set of properties.

Cell name	NMOS
Symbol name 1	N1
Symbol name 2	N2
Schematic name	Schematic

Scenario 5

You want to have a single symbol of an amplifier with a basic and a high precision implementation of the schematic so you can switch between the two to trade off accuracy and run time in different simulations. (The high precision schematic could contain, for example, additional parasitic information back annotated from layout.) You would create a cell with a single interface, a single symbol, and two schematics.

In a case where there are two schematics and a single symbol, there is an ambiguity in which schematic to use. S-Edit will use the schematic whose name matches the symbol name. To switch from using one schematic to another, use **Cell > Rename View** to change the symbol name to match that of the desired schematic. In this example, to switch from using the basic schematic to using the precision schematic, you would change the symbol name from “Amplifier_basic” to “Amplifier_precision.”

Cell name	Amplifier
Interface name	Interface
Symbol name 1	Amplifier_basic (or Amplifier_precision, depending on the schematic you want to use.)
Schematic name 1	Amplifier_basic
Schematic name 2	Amplifier_precision

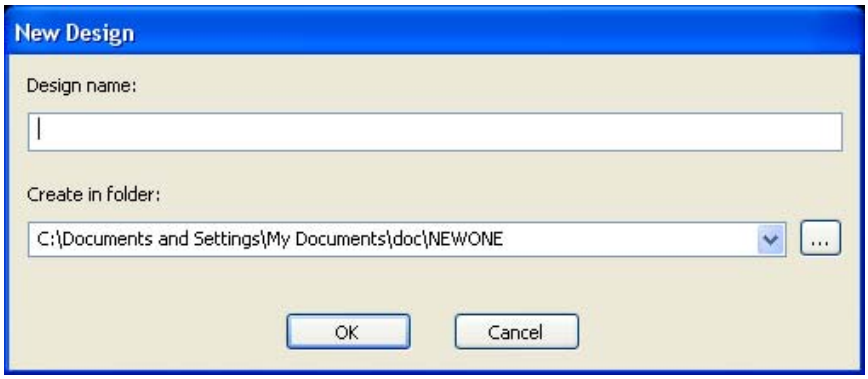
Creating a Schematic Project

An S-Edit project consists of the design itself, plus any number of libraries. The design contains cells, which may reference other cells in the same design or may reference cells in libraries. A reference to cell “A” consists of the instancing of a symbol of cell “A” within the schematic of another cell. A library is simply a design that is referenced by another design.

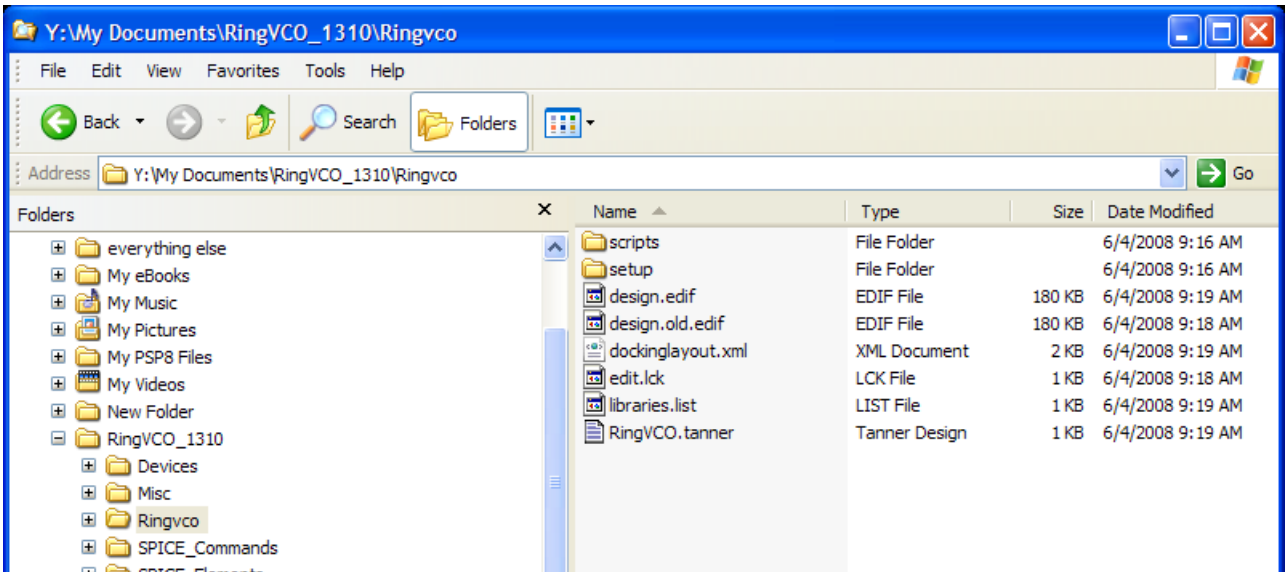
A library must be added to a list of libraries available to a design before its cells can be referenced. When you open a design, S-Edit will also open all libraries that are referenced by that design. There is no functional difference between a library and a design, so in the Libraries navigator, the design name is shown in black and library names are shown in blue.

Creating a New Design

Use **File > New > New Design** to enter the name and directory location for a new design.



Project File Structure in S-Edit



S-Edit stores design information in several different files in the {*designname*} directory.

- **design.edif** is the design itself.
- **design.old.edif** is a backup of the design file. It is overwritten each time the design is saved.
- **dockinglayout.xml** stores any previously used (or else the default) workspace settings.
- **edit.lck** is a lock file that prevents an open design from being opened elsewhere.
- **libraries.list** references the libraries used in the design. Library path names in this file can be edited if desired.
- **{*designname*}.tanner** is a TCL file that launches a design. It specifies the path, cells, and windows to open, and references other necessary files. For example:

```
workspace loadfrom -path .
cell open -design Ringvco -cell RingVCO_TestBench -type schematic -newwindow
window move -left 224 -top 224 -width 1630 -height 573 -undock -max
```

```
window fit -x0 -4425 -y0 -2290 -x1 16470 -y1 9208 -units iu
workspace dockinglayout load -file {./dockinglayout.xml}
source open.design -relativeto user
source open.design -relativeto design
```

- The **setup** folder stores any changes that have been saved from the Setup dialog (see "Saving and Loading Setup Options" on page 35).
- The **scripts** folder contains the three default subdirectories for scripts that S-Edit reads automatically when it is launched, closed and when a design is opened (see "Executing Scripts Automatically" on page 203).

Path Name Syntax and Delimiter Evaluation

Warning: All fields in S-Edit are evaluated. Because the back slash character (****) is an escape-sequence prefix in S-Edit, you can no longer use a single back slash as a path separator.
Path separators should be either a forward slash (**/**) or a double back slash (****).

For the same reason, S-Edit automatically translates back slashes to forward slashes when it opens version 13 or 12 designs, and when you press the **Browse** button from a file or directory name field.

If you enter a back slash, the expression evaluator will automatically remove it, unless the character it precedes triggers a special function. For example:

```
Y:\My Documents\EDAprojects\New\18micron\RingVCO_TestBench.sp
```

S-Edit returns

```
Y:My DocumentsEDAprojectsNew18micronRingVCO_TestBench.sp
```

When you have a special escape sequence, S-Edit will interpret it. For example, the escape sequence **\t** yields a tab space. If you type:

```
Y:\My Documents\doc\2008\test.sp
```

as an include file, S-Edit returns the following in the exported netlist:

```
.load file="Y:My Documentsdoc2008est.sp"
```

A very few special fields provide syntax checking.

Creating a New Text File

Files in the context of S-Edit menus are always text files, typically of the type for which the S-Edit text editor provides syntax highlighting (C, log, SPICE, EDIF, etc.). Use **File > New > New File** (**Ctrl + N**) to open the text editor in the design window.

Environment Variable Usage in File Paths

All Tanner applications allow the use of environment variables in file paths. An environment variable named VARIABLE may be referenced in three ways:

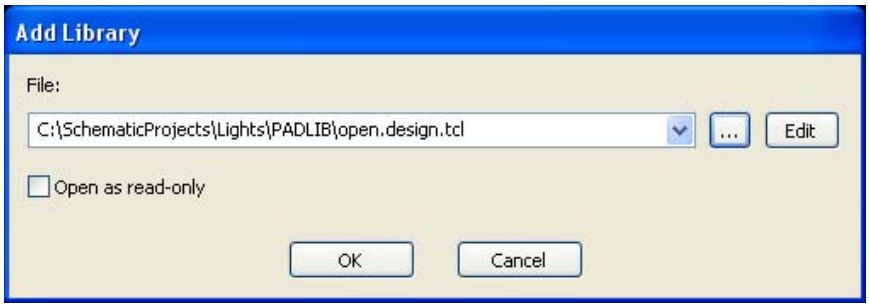
- %VARIABLE%

- $VARIABLE
- ${VARIABLE}

Adding a Library

When you load an existing design, any libraries referenced by that design are also loaded and will appear in the Libraries navigator. When you create a new design, or if you want to reference cells in a library that is not currently open, you will need to explicitly load that library.

Use either **File > Open > Add Library** or the **Add** button in the Libraries navigator to open the dialog shown below.



When you add a library, S-Edit will also load the other libraries it references.

Library locations are saved to the **libraries.list** file. For portability, they are saved with a relative path if the library is in either a subfolder of the design or a descendant folder of the design. Libraries that are in folders above the level of the design are saved with an absolute path.

Absolute paths in **libraries.lst** are preserved when the design is saved.

Opening a Design

Each design in S-Edit has a script file named "***filename*.tanner**" that opens the design and specifies the related libraries, interface and related project settings.

Use **File > Open > Open Design** (shortcut **Ctrl+O**) to open an existing design. Enter the path to the design under **File**, select from the drop-down menu, or use the ... browse button to search for the folder with the ***filename*.tanner** file for the design you want to open.

There are several other ways to open a design. From a browser you can drag and drop a **.tanner** file into the S-Edit Command Window, or you can double-click on a ***filename*.tanner** file. You can also select from the **File > Recent Files** list. Each of these methods will attempt to open the design with "exclusive access" (see "Exclusive vs. Non-Exclusive Access to Designs," below.)

When S-Edit opens a design, scripts are read in this order:

- Libraries in the libraries.list file are loaded, using depth-first recursion (described below)
- Setup scripts are read from the design folder.
- Setup scripts are read from the user preferences folder.
- A source statement in the design script runs scripts in the "open.design" folder relative to user settings.

- A source statement in the design scripts runs scripts in the “open.design” folder relative to design settings.

When a library is opened because it is listed in a design’s libraries.list file, scripts are read in this order:

- Setup scripts from the library's design folder. [...\libraryname\setup]
- Setup scripts from the user preferences folder. [...\username\scripts\open.design\setup]
- All the scripts in the user’s preferences [...\username\scripts\open.design]
- All the scripts in the design’s settings [...\libraryname\scripts\open.design]

Scripts corresponding to setup dialog settings have special names. Putting other scripts in the folders where they are saved has no effect. When all scripts in a folder are executed, as with the open.script folder of a design, they are executed in alphabetical order of file name.

Exclusive vs. Non-Exclusive Access to Designs

“*Exclusive access*” means reserving the right to save a design so that no-one else can write to it while you have it open. This “*write reservation*” gives you the exclusive right to save a design.

“*Non-exclusive access*” means you *might* be able to write to the design in the future, as long as no one else secures a write reservation before you attempt to save the design. If someone else has exclusive access to a file but then releases it by closing the file, you will be able to write to that file.

Note that exclusivity is independent of the **Allow Editing** setting.

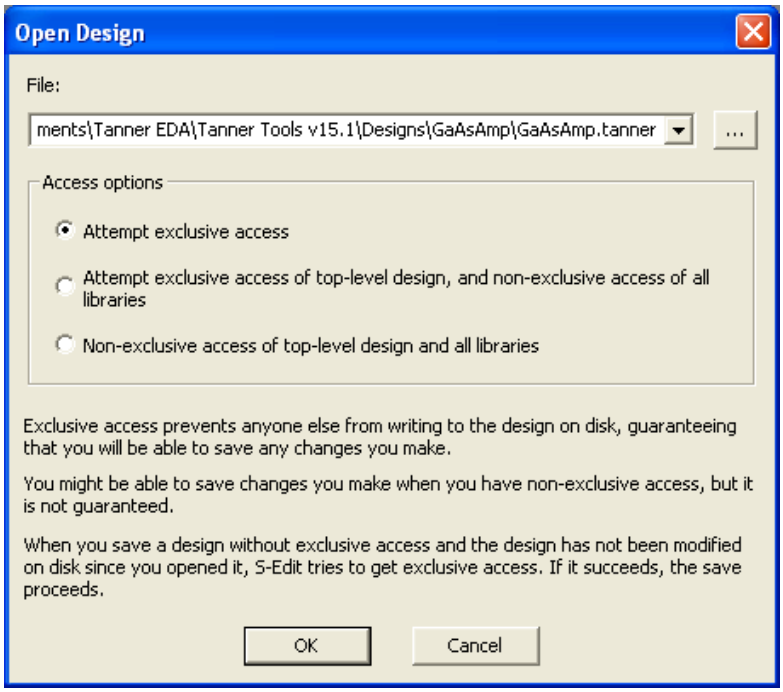
If a design is opened with exclusive access and editing is allowed, no Library Navigator icon is displayed.

If a design is opened with exclusive access and editing is not allowed, the icon is a yellow lock ( ).

If a design is opened with non-exclusive access and editing is allowed, the icon is a red stick figure ( ).

If a design is opened with non-exclusive access and editing is not allowed, the icon is a red lock ( ).

Note that failure to obtain exclusive access to a library is silent, so the first two **Open Design** options can appear to have the same results.


Open Design
File:
ments\Tanner EDA\Tanner Tools v15.1\Designs\GaAsAmp\GaAsAmp.tanner
...
Access options
Attempt exclusive access
Attempt exclusive access of top-level design, and non-exclusive access of all libraries
Non-exclusive access of top-level design and all libraries
Exclusive access prevents anyone else from writing to the design on disk, guaranteeing that you will be able to save any changes you make.
You might be able to save changes you make when you have non-exclusive access, but it is not guaranteed.
When you save a design without exclusive access and the design has not been modified on disk since you opened it, S-Edit tries to get exclusive access. If it succeeds, the save proceeds.
OK
Cancel

S-Edit checks for and warns about file modifications. For example, if user A opens a design exclusively, and user B opens the same file subsequently, user B will not be able to save edits to the file. If user A edits the file, saves it and exits S-Edit, at that point the write protection is released. However, user B will not be able to make changes or to save the file successfully since it has been updated since user B first opened it.

Attempt exclusive access

When you open a design “exclusively,” S-Edit places a “write reservation” on it that reserves the right to save that design in the future, for as long as you hold the write reservation by keeping the file open.

You are reserving the right to write to that design in the future. When you hold the write reservation, nobody else can have the same right.

This is the default.

Attempt exclusive access of top-level design, and non-exclusive access of all libraries

This option places a write reservation on the top-level design only, allowing others to have a write reservation during your edit session on the libraries it uses.

Note that failure to obtain exclusive access to a library file is silent - you will not be notified that you do not have a write reservation for a library.

Non-exclusive access of top-level design and all libraries

Does not secure a write reservation for any file, but you can save any file you’ve opened as long as no one else takes the write reservation it or changes it.

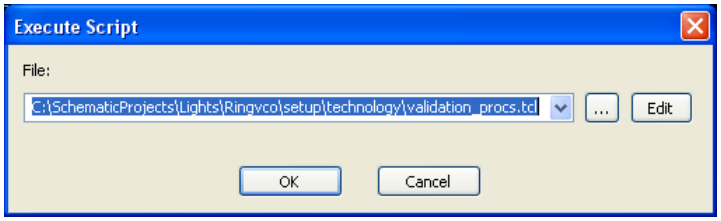
Opening a TCL File

Opening a TCL File for Editing

Use **File > Open > Open File** to open a TCL file in the S-Edit text editor without executing it, or to open any other kind of text file. As of v14, Tanner Tools support TCL v8.5.

Executing a TCL File

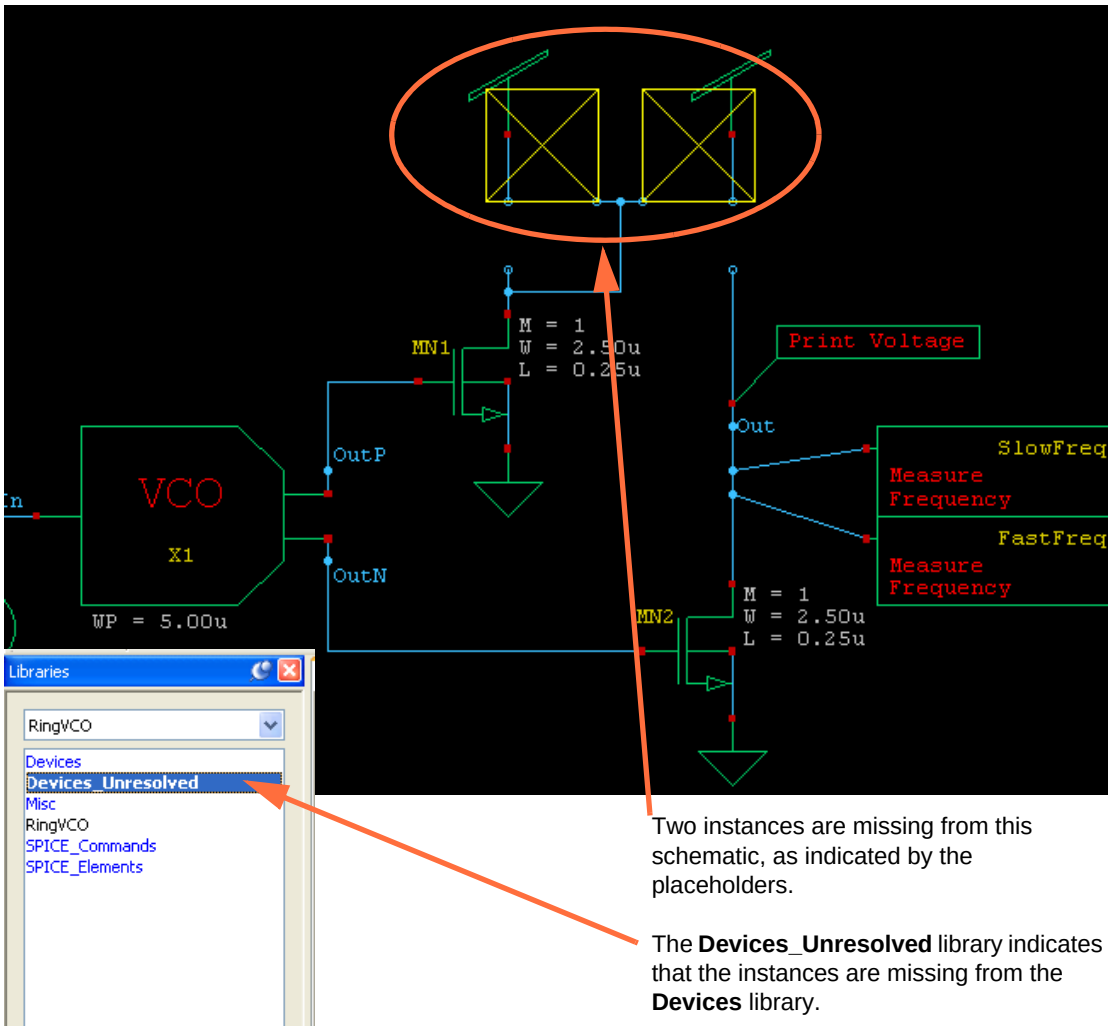
Use **File > Open > Execute Script** to find and execute TCL file in the Command window. You can also drag a TCL file icon from a browser and drop it into the Command window to run it.



Cells or Libraries Missing From a Design: _Unresolved

Occasionally a cell or even an entire library may be missing when you open a design. This can happen when cells in a library are updated or renamed, when a referenced library is relocated or deleted, or when cells are lost or corrupted during an import process.

If cells are missing from a design, S-Edit adds a new library with "**_Unresolved"** appended to the name of the library from which the cell is missing. This **_Unresolved** library lists the missing cells, drawn as crossed-out boxes which are displayed as placeholders in the schematic.

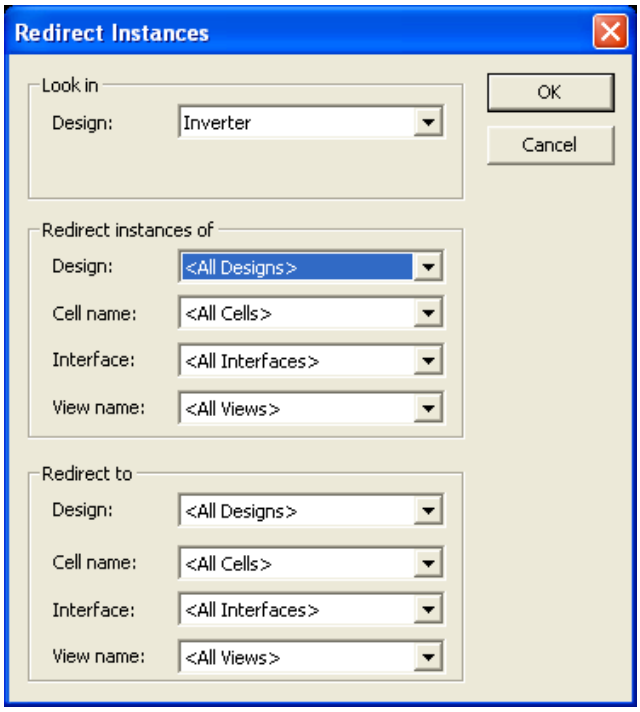


Two instances are missing from this schematic, as indicated by the placeholders.

The **Devices_Unresolved** library indicates that the instances are missing from the **Devices** library.

Resolving Missing Cells or Libraries

Use **Cell > Redirect Instances** when a cell or library is missing from your design.



Look In — Select the design in which the cell reference will be redirected.

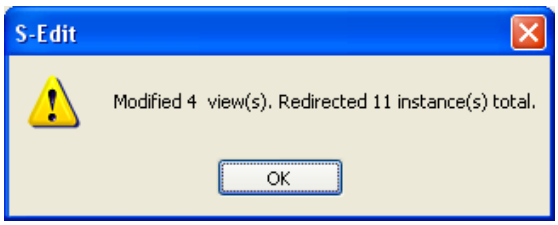
Redirect instances of — Select from these drop-down lists to specify the design, cell, interface type and view name that you want to replace.

- **Design**
- **Cell Name**
- **Interface**
- **View Name**

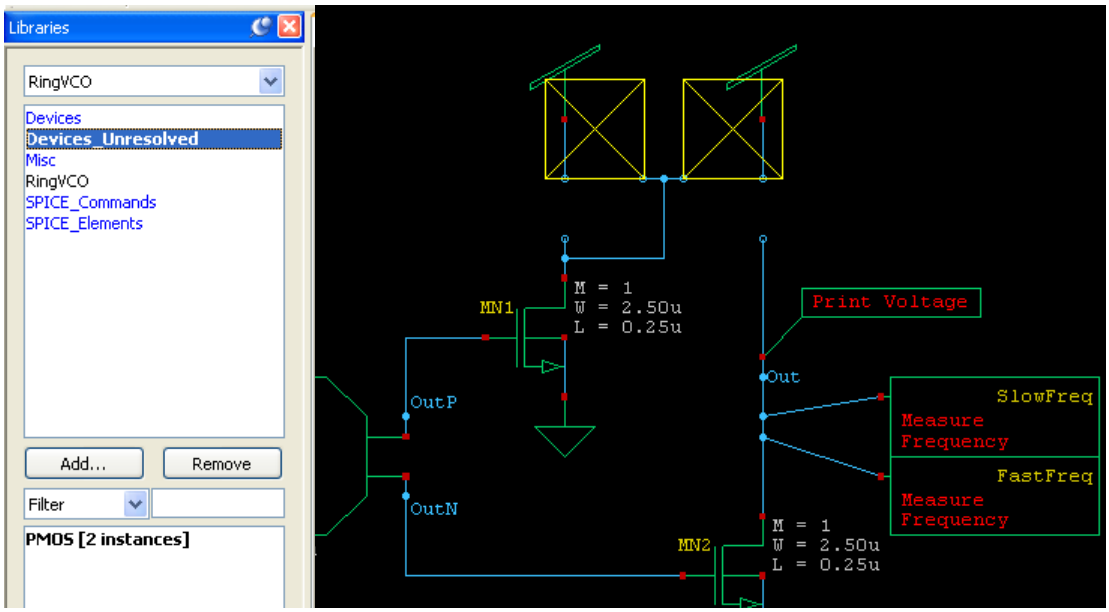
Redirect to — Select from these drop-down lists to specify the design, cell, interface type and view name that should be used in the active design.

- **Design**
- **Cell Name**
- **Interface**
- **View Name**

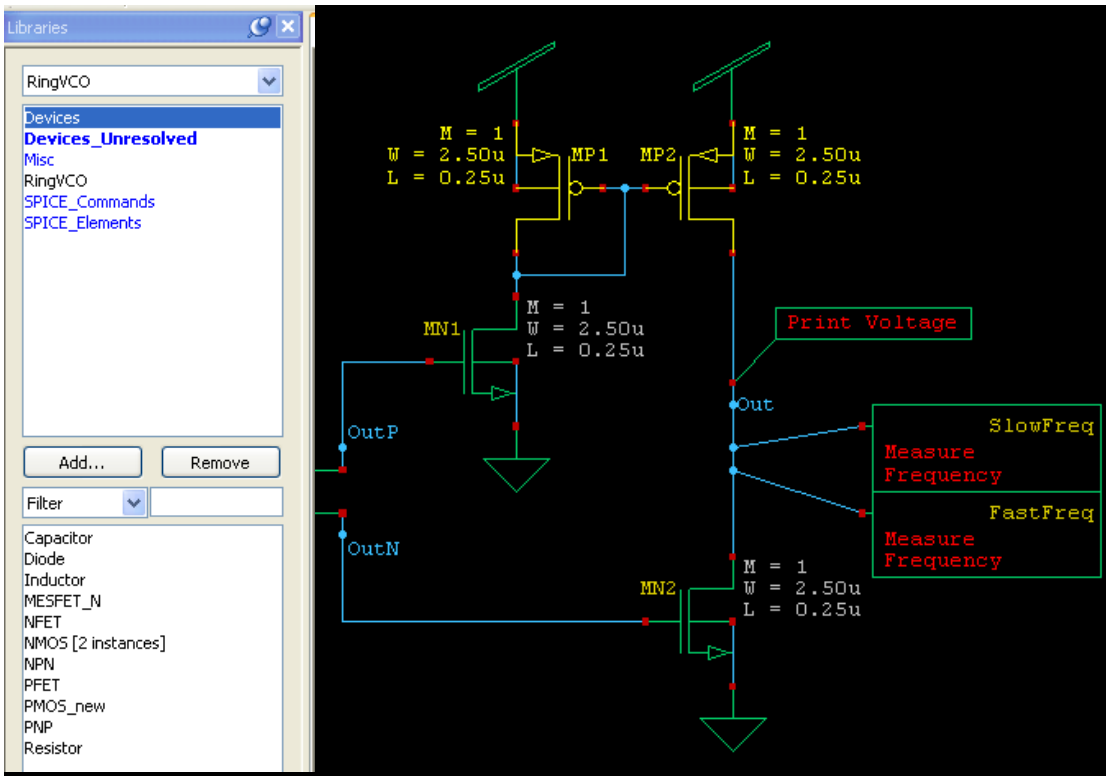
S-Edit will confirm the number of views and instances updated.



In the design **RingVCO** shown below, the library **Devices_Unresolved** contains placeholders for the missing cell **PMOS** (highlighted in yellow). The library **Devices** is missing these two cells.



When you use **Cell > Redirect Instances,** you are redirecting the reference in design **RingVCO** so that the proper cell, in this case cell **PMOS_new,** which should be in the library **Devices,** will replace the previously referenced cell **PMOS**, which S-Edit has moved, in placeholder form, to the temporary library **Devices_Unresolved**.



Closing a Design

When you close an S-Edit design file, the document is removed from memory. To close a design, use the **File > Close > Close Design [filename]** command.

Saving a Design or Library

Use the **File > Save > ...** commands to save a design, the libraries it uses, or a combination of these.

When multiple designs are open, the one in the currently active window will be saved. (The design name is displayed in the save menu for confirmation.) S-Edit creates a "**design.old.edif**" backup file each time a design is saved.

If a design or library is set so that edits are not allowed, S-Edit will show a message in the command window alerting you that the save operation has not been successful.

Please also refer to the section "Exclusive vs. Non-Exclusive Access to Designs" on page 63 for a description of how the authority to edit or save changes to a file is controlled, and to the section "Setup Values by Design versus Library" on page 37 for a description of active files and the **Allow Editing** setting.

Save Design [*filename*] — Saves the active design only and does not save changes to libraries.

Save all changes — Saves the active design and any of its libraries that have changed since the last save operation.

For read-only libraries, S-Edit will attempt to get a write lock and, if successful, will save the library. If not successful, S-Edit will display a message in the command window to that effect, such as "**# SED Error: Edits to design "{*designname*}" are not allowed.**" or "**# SED "Library not saved because it has not been modified.**"

Save [*number*] Selected Design/Libraries — Saves only those designs or libraries selected in the Libraries navigator. For confirmation, the number of files selected is dynamically updated in the save menu.

Save Copy of [*number*] Selected Design/Libraries — Saves a copy of the selected designs or libraries to the specified new location.

Note: Once the files are copied, S-Edit returns to the file *that was active prior to the command.*

Save Copy Of

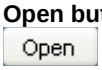
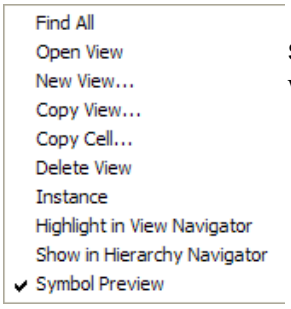
File > Save > Save Copy of ... is different than the "**save as**" command. After the **Save Copy of ...** operation, S-Edit returns to editing the original design, NOT the newly saved copy.

You must use the "**Save Copy of Design/Libraries**" command to be able to rename a design. Simply renaming the directory does not change the name of a design or library.

Cell and View Operations

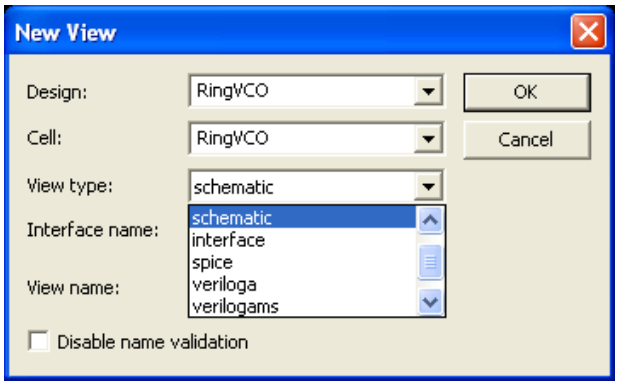
S-Edit has a highly flexible interface that offers many ways to perform key operations besides the traditional menu options such as **Cell > Open**. In addition to the standard menu shortcuts, keyboard shortcuts, and customized toolbars, you can use the following different methods to perform cell and view operations.

Shortcuts for Cell and View Commands

Operation	*Interface: Action*	*Behavior*
◆ **Open a cell**	Libraries navigator: **Double-click**	Highlight a cell in the list and **double-click** to open it. The view type opened will be the same as what was last opened.
◆ **Open a cell**	Libraries navigator: **Open button** 	Highlight a cell in the list and press the **Open** button to open it in schematic view.
◆ **Open a cell**	**Design area: Double-click**	Select a cell in the design area and double-click to open it in schematic view.
◆ **Instance a cell**	Libraries navigator: Click-and-drag	Highlight a cell in the list, click and drag it into the design area to instance a cell. Double-click to place it once you have chosen a location.
◆ **Open a view**	Libraries navigator: drop-down menu	Highlight a cell in the list and use the drop-down arrow next to the **Open** button to select one of its views to open.
◆ **Find All** ◆ **Open View** ◆ **New View** ◆ **Copy View** ◆ **Copy Cell** ◆ **Delete View** ◆ **Instance** ◆ **Highlight in View Navigator** ◆ **Show in Hierarchy Navigator** ◆ **Symbol Preview**	Libraries navigator: **Right-click** and **select from menu** 	Highlight a cell in the list, **right-click** to open the context-sensitive menu, and click on the desired operation. **Symbol Preview** shows or hides the preview window in the Libraries navigator.

Creating a New View (Cell > New View or “N”)

To create a new cell or a new view of an existing cell, use **Cell > New View** (shortcut **N**). Simply select from the drop-down menus in each field.

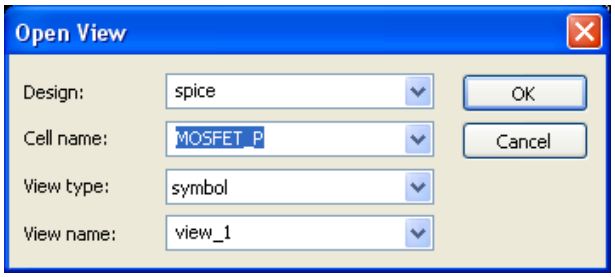


Design	Select a file to which the cell will be saved from the drop-down list of designs and libraries open.
Cell	Enter or select a cell name.
View type	Select a view type for the cell from the drop-down list: symbol, schematic, interface, spice, veriloga or verilogams. A new Verilog-A view will be initialized with the module name. If the new view is attached to an existing interface view, the Verilog-A view is also initialized with the list of ports.
Interface name	Select the interface view to which the new cell will belong.
View name	Enter a view name for the cell.
Disable name validation	Check this box to prevent S-Edit from checking for naming violations.

Opening a View (Cell > Open View or “O”)

The **Cell > Open View** command (shortcut **O**) lets you open a cell view from any of the designs currently loaded in S-Edit.

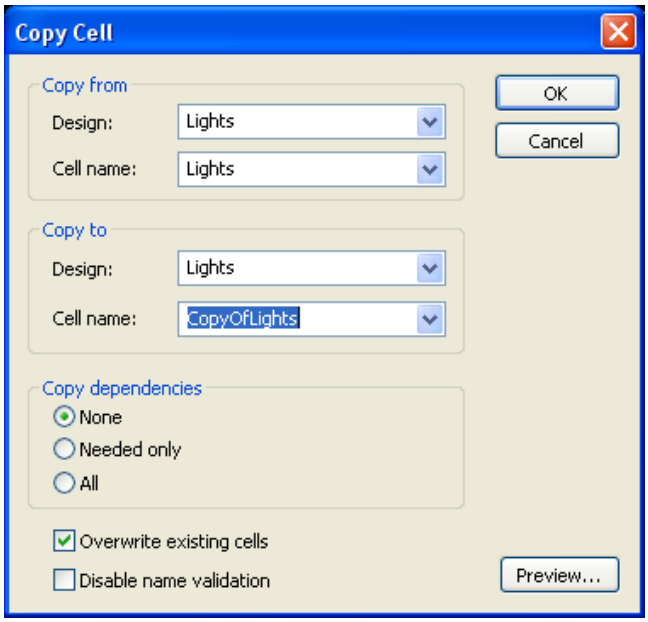
Note that if you have name validation enabled and you want to open a cell or view that has an invalid name, you will have to disable name validation from the **Setup > Technology > Validation** page.



Design	Select a design file or library file from the menu of those open in S-Edit.
C**ell name**	Select the cell to open from the drop-down menu.
View type	Select the type of view to open.
View name	Select a specific view to open.

Copying a Cell

Use the **Cell > Copy Cell** command to duplicate a cell and all its views. Cells can be copied within the current design or from one design to another.



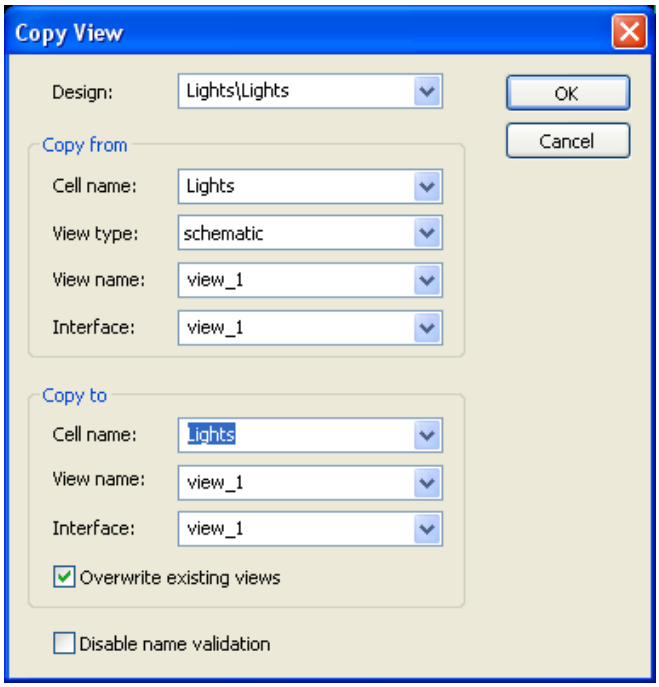
Copy from	**Design:** select a source design or library file from the menu of those open. **Cell name**: select a source cell from the drop-down menu.

Copy to	**Design**: select a destination design file from the list of those open. **Cell name**: enter a name for the new cell. S-Edit will preprend "CopyOf" to the cell name by default if you do not enter a name.
Copy dependencies	**None:** only the select cell is copied. **Needed only**: only the instanced cells that do not exist in the destination file are copied. **All**: all instanced cells within the cell are also copied, down through the hierarchy. Note that S-Edit only checks for cell name during this operation, not cell contents or save date.
Overwrite existing cells	When checked, the copy will overwrite the existing cell(s) of the same name.
Disable name validation	Check this box to prevent S-Edit from checking for naming violations.

Copying a View

Use the **Cell > Copy View** command to copy a single view of a cell to the same or a different target cell. A view can only be copied within the same library of a design. See the "How to Use Multiple Views" on page 57 for some of the ways it can be useful to define multiple views of a given cell.

You can use this dialog as a shortcut to creating a new cell by entering a new name in the target **Cell name** field. Note that if you enter the name of an existing cell with the overwrite option active, if all other view names are the same as in the existing cell, the existing view will be replaced.



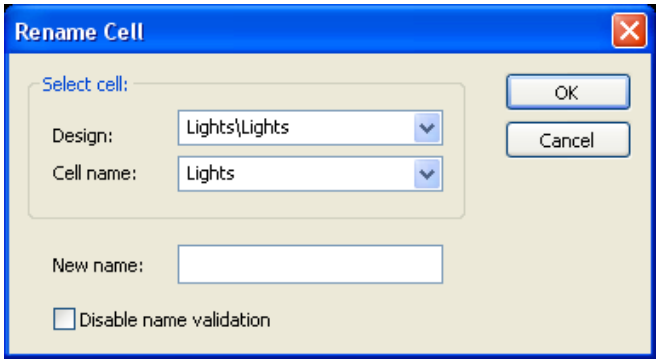
Design	Select a source design or library file from the menu of those open.
C**opy from**	Select a **Cell name**, **View Type**, **View name**, and **Interface** to copy. S-Edit will preprend “CopyOf” to the cell name by default if you do not enter a name.
C**opy to**	Select a **Cell name**, **View Type**, **View name**, and **Interface** for the new view from the pulldown menu, or enter new names.
Overwrite existing views	When checked, the copy will overwrite the existing view of the same name. When unchecked, if the view already exists, will not change it. In either case, if the view does not exist S-Edit will create it.
Disable name validation	Check this box to prevent S-Edit from checking for naming violations.

Instancing a Cell

Instances in a cell are references to other cells. These cells might be common library cells (such as basic circuit elements) or larger, custom-designed cells. Instances are dynamically linked to their source cell so that any change you make to a source cell is reflected in each higher-level instance of that cell. (See “Creating Instances” on page 115.)

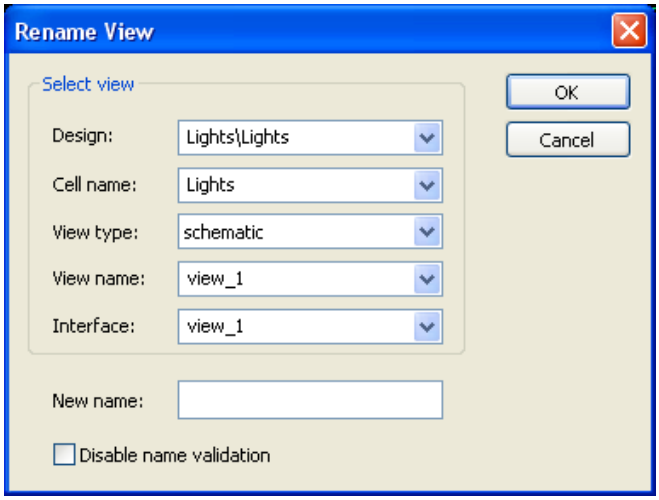
Renaming a Cell

The **Cell > Rename Cell** command will rename a cell but not its views. This operation keeps the renamed cell open.



Renaming a View

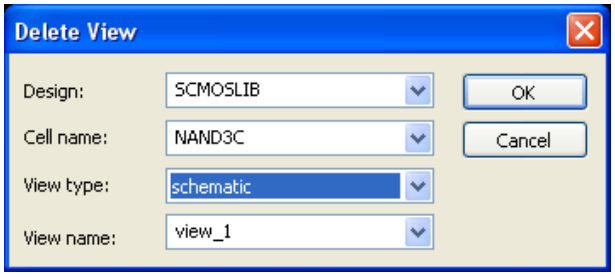
Use the **Cell > Rename View** command to rename a view.



Deleting a View

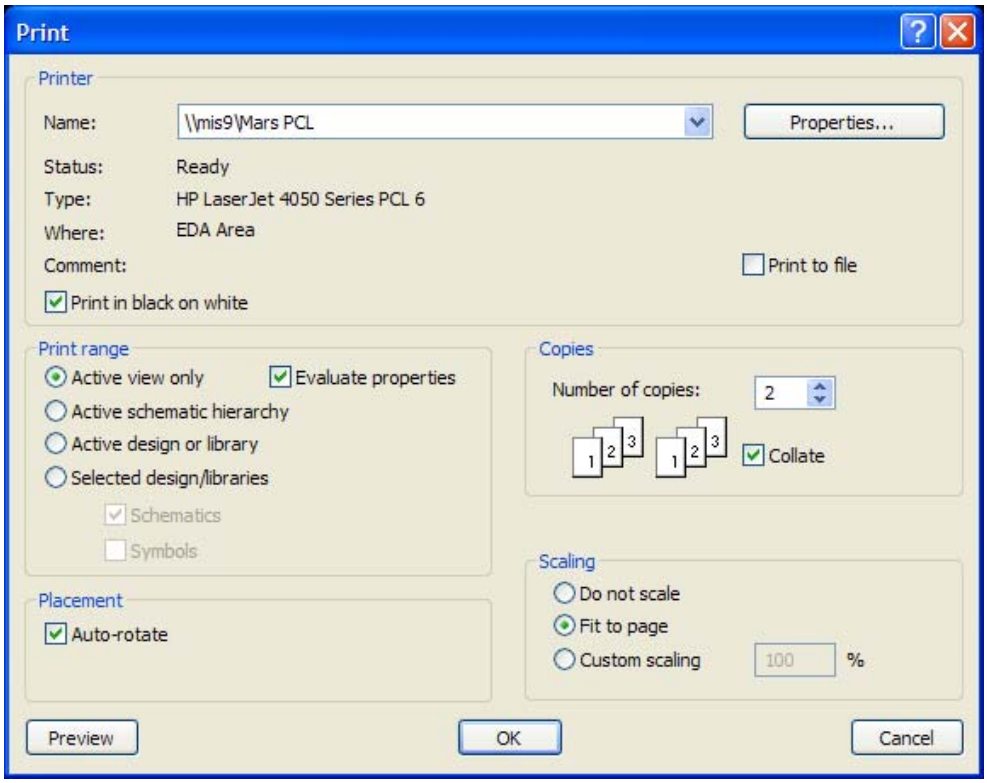
The command **Cell > Delete View** deletes a specified view, or all views of a cell. When you issue this command, S-Edit displays the **Delete View** dialog box, which prompts for the name of the cell to be deleted. A symbol view or its interface command cannot be deleted if the symbol is instanced.

When the last view of a cell is deleted the cell itself is also deleted.



Printing a Design

Use **File > Print** to control the scope and scale of your printouts. Use **File > Page Setup** to specify the layout of printed pages (see “Setting Page Size” on page 42).



Printer — **Name**, **Type**, **Where**, and **Comment** all identify the active printer. **Status** describes the state of the selected printer—busy or ready.

Properties — Opens the **Printer Properties** dialog for additional printer properties.

Print in black on white — If this is not checked, the design will print just as it appears on screen.

Print to file — When this is checked, S-Edit creates a .PRN format file to the location you specify.

Print Range

- **Active view only** prints just the current view.
- **Active schematic hierarchy** prints the current view and all schematics in the hierarchy beneath that view.

With the following two options, you can elect to print just **Schematics** or just **Symbols**:

- **Active design or library** prints all pages of the active design
- **Selected design/libraries** prints the design file that is highlighted in the Libraries navigator, regardless of the active view.

Evaluate properties

Check this box so that properties will print in evaluated form. This option applies only when printing the active view. (If properties are currently evaluated, the option has no effect.) The default setting is on (checked).

The PageID cell that ships with S-Edit uses special system properties, see "Printing a Title Block," below.

Copies

Enter the number of copies to print.

Collate

When checked, prints copies of a file in proper page order. If not checked, each copy of the first page will be printed consecutively, then the second page, etc.

Placement

When **Auto-rotate** is checked, the image will be rotated to best fit the selected page size and scaling.

Scaling

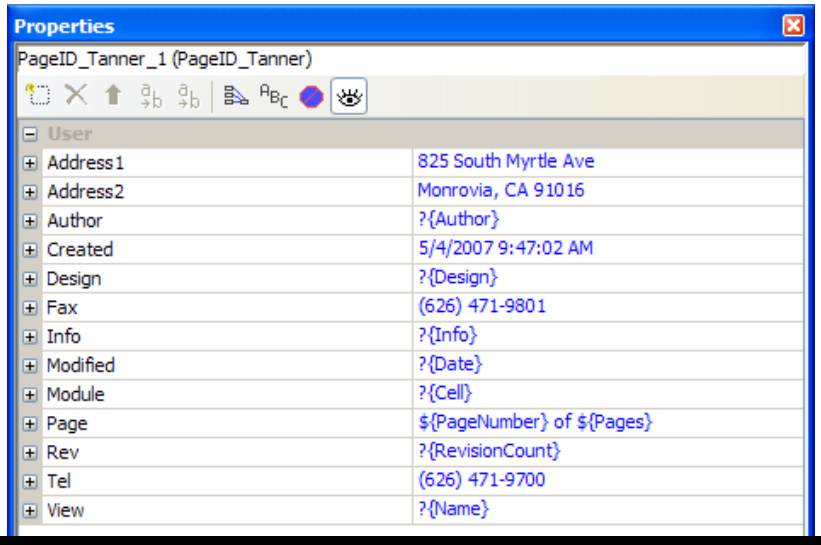
Do not scale prints the design as set in **Setup > Technology > Schematic Page**.

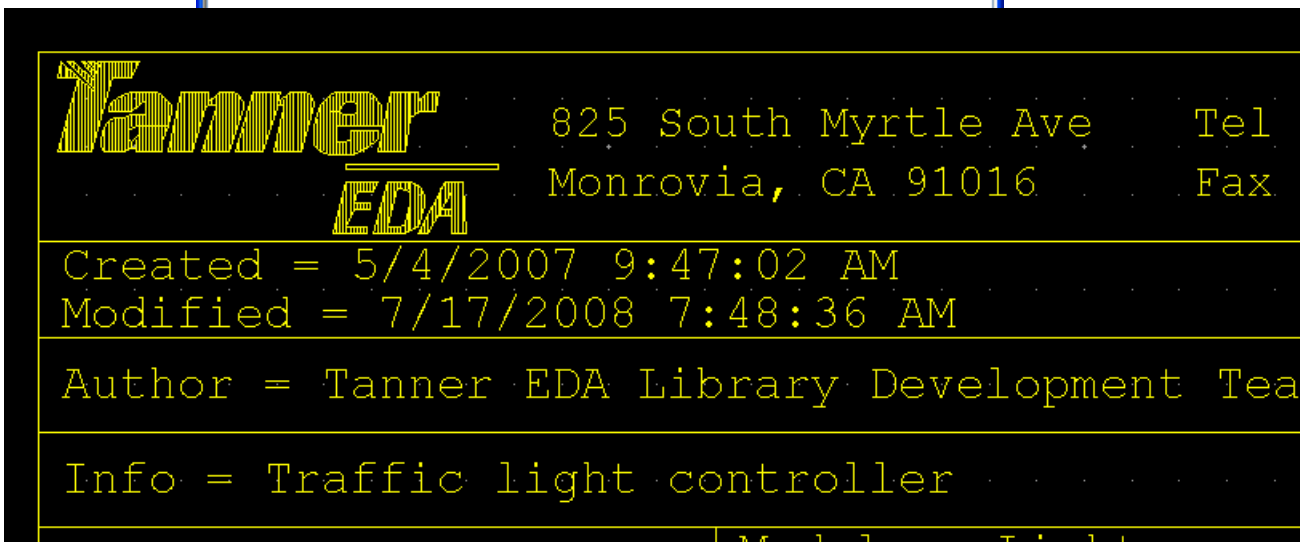
Fit to Page scales the design to fit the paper size on which it will be printed.

Custom scaling lets you use percentage values to set the size of the printout with respect to the page layout set in **Setup > Technology > > Schematic Page**.

Printing a Title Block

The **PageID** cell included with S-Edit in the **Misc** library uses the following evaluated system properties, as well as several user properties. By default, evaluated properties are set to on during printing



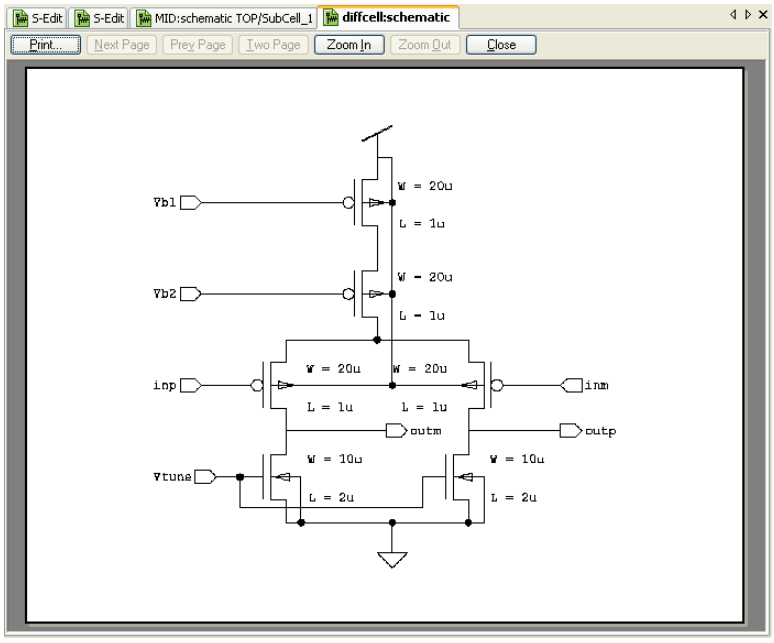


?{Design}	Retrieves the read-only value from the **Design** name field, a system property in the schematic view that contains the instance.
?{Date}	Retrieves the read-only value from the **Date** (last modified field), a system property in the schematic view that contains the instance.
?{Cell}	Retrieves the read-only value from the **Cell** name, a system property in the schematic view that contains the instance.
${PageNumber}	${PageNumber} evaluates to the 1-based index of the page that contains the instance.
${Pages}	${Pages} evaluates to the total number of pages in the schematic view that contains the instance.

?{RevisionCount}	Retrieves the read-only value from the **Rev** system property in the schematic view that contains the instance. This value increments by one each time the file is modified.
?{Name}	Retrieves the read-only value from the (view) **Name** field, a read-only system property in the schematic view that contains the instance.
Created	A read-only field displaying the date and time the file was created.
?{Author}	Retrieves the user entry from the **Author** field in the schematic view that contains the instance.
?{Info}	Retrieves the user entry from the **Info** property field in the schematic view that contains the instance.
Organization	Retrieves the user entry from the **Organization** field in the schematic view that contains the instance.
Address 1 & 2	Retrieves the user entry from the **Address1** and **Address2** fields in the schematic view that contains the instance.

Print Preview

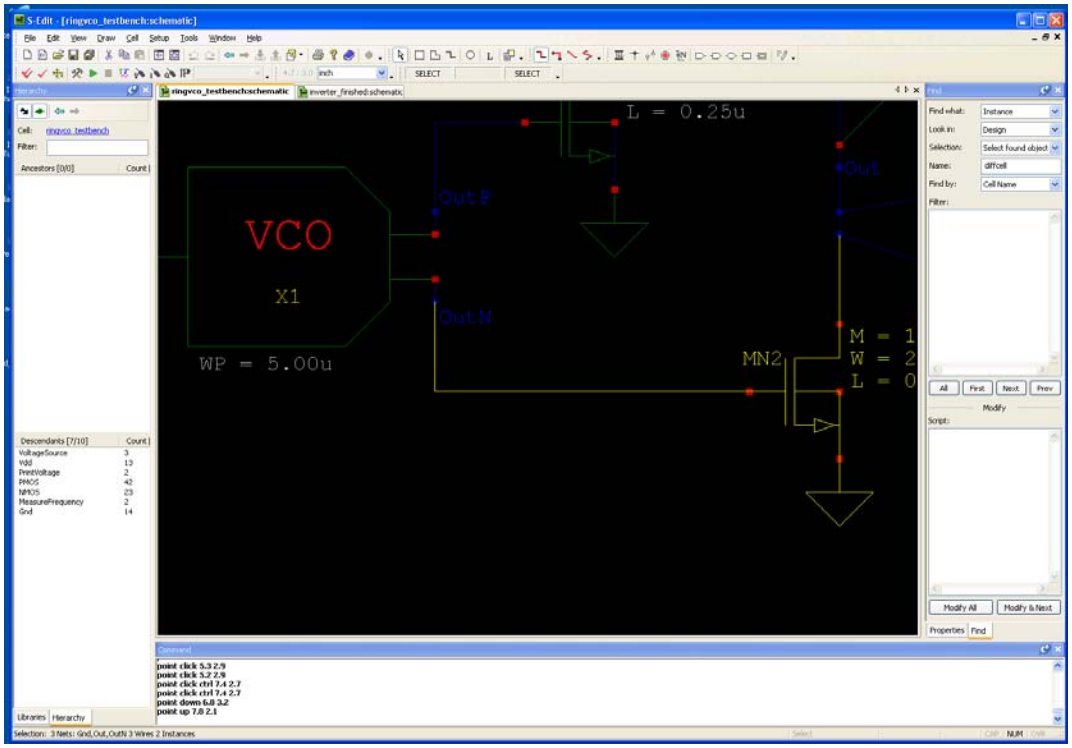
File > Print Preview lets you preview the active window view in the S-Edit design window. Unless **"Print in black on white"** (page 76) is checked, S-Edit will print using the display colors.



3 Navigating and Viewing a Design

The Work Area

The visible portion of the design area of a design, where you create, view, and edit objects, is called the *work area*. You can move or pan the work area to show a different part of the design area, or change its magnification to show a larger or smaller part of the design.



When you resize the work area, S-Edit expands or contracts the view in the direction in which you resized, without changing the magnification.

S-Edit's pan and zoom operations are always *active*: you may pan or zoom while drawing, moving, or editing an object—and *interruptible*: once you have initiated a panning or zooming operation, you need not wait for the screen to redraw completely before you initiate another one.

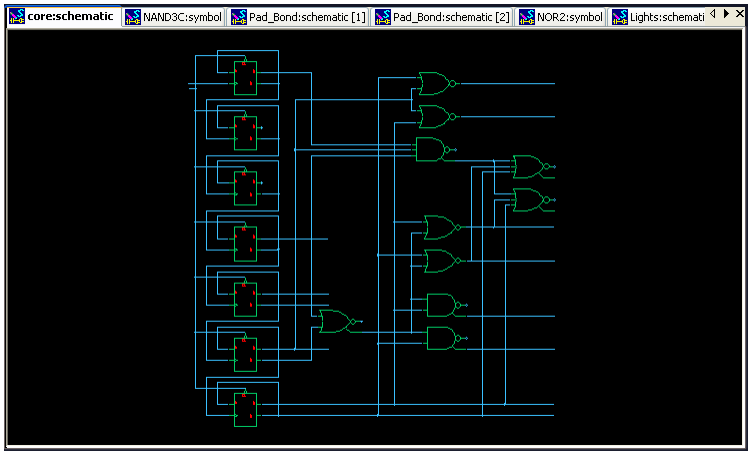
Opening Design Windows

You can open multiple design windows in S-Edit. Design windows can be tiled or cascaded, but the default view is as tabs. If you right-click in the tab area you can use the context-sensitive menu there to

"**Close All**" open views or "**Switch Tab Placement**" from the top to the bottom of the window and vice-versa.

Use the arrows in the upper right corner to scroll to open windows that are offscreen. A filled-in arrow indicates the presence of offscreen windows, otherwise is it outlined.

When you maximize one design window, they will all be maximized. You can right-click directly on a tab to open a context-sensitive menu to "**Close**" that tab or make it "**Dockable**."



Reusing Design Windows

You can also control how S-Edit reuses design windows to avoid having too many windows open at once.

By default, each time you open a new view or cell, S-Edit opens it in a new window. You can keep this default behavior, or you can use **Setup > Preferences > General** to change the default so that S-Edit opens only one window at a time, regardless of the view type. For example, if you have a schematic view of cellA and you open a symbol of cellB, S-Edit will replace the cellA display with cellB, reusing the window that is open.

You can force a new window to open at any time by holding the **Ctrl** key down while opening a view.

Changing Windows and Views

Redrawing the Screen

You can refresh display in the active window at any time by using **View > Redraw** (shortcut **Space**).

Cycling Focus Between Windows

Use the **View > Goto > Previous** and **View > Goto > Next** commands to cycle focus through each pane that has been active in the current editing session.

You can also use the **Backward** ⇦ and **Forward** ⇨ icons to step backwards or forwards through the previous and next views that were opened. Note that the **Forward** button is only available after stepping **Back**.

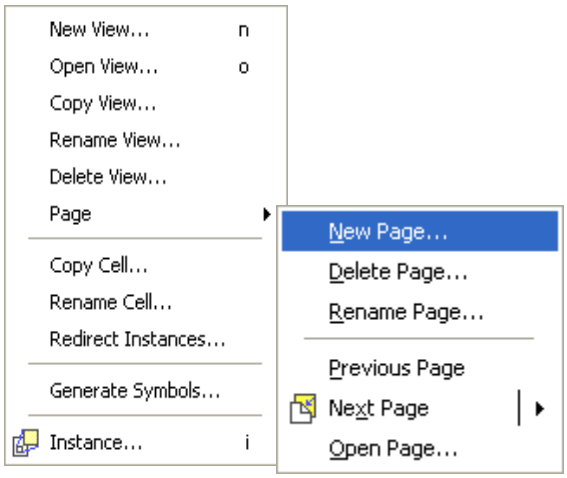
Cycling Views Within a Window

Use **View > Exchange** (shortcut **X**) to return to the previous view within a given pane after you execute any zoom or pan command. You can use this command to toggle back and forth between two views. If you return to a previously stored view after resizing the work area, S-Edit will alter the aspect ratio of that view to fit the current window.

Use **View > Cell View > Symbol** and **View > Cell View > Schematic** to select one of these view types for the active cell. Or, use **View > Cell View > Cycle View** to cycle through all the views for a given cell.

Splitting a Cell into Multiple Pages

S-Edit allows you to split a schematic across multiple pages. Use **Cell > Page** to add, remove or rename pages, scroll backwards and forwards though pages, or open a page.



The **Next Page** toolbar button cycles through a cell's pages as well as providing a drop down menu with the option to select a specific page or create a new page.

Showing and Hiding Grids

You can use **View > Display > {Display Major Grid, Display Minor Grid or Display Origin}** to display, or not, each of these elements. This setting applies just to the active design.

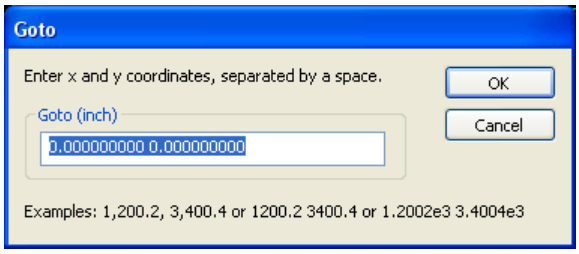
Panning

You can pan to different portions of the design area by using the arrow keys, or one of the **View > Pan** commands.

Command	*Shortcut*	*Function*
View > Pan > To Selection		Centers the view over the selected objects. Depending on the magnification, all selected objects may not be visible in the resulting view.
View > Pan > Left	(left arrow key)	Moves the work area to the left by one quarter of the width of the display.
View > Pan > Right	(right arrow key)	Moves the work area to the right by one quarter of the width of the display.
View > Pan > Up	(up arrow key)	Moves the work area up by one quarter of the width of the display.
View > Pan > Down	(down arrow key)	Moves the work area down by one quarter of the width of the display.
View > Pan > To Edge **Left Edge** **Right Edge** Up Edge Down Ege		Moves the display window in the direction you indicate, to the farthest edge of all objects in the design area.

Panning to a Specific Location

Use **View > Goto > Coordinates** to center the view on a specific coordinate. Enter x, y coordinates in display units, as measured relative to the origin, separated by a space.



Zooming

You can zoom in S-Edit using the mouse buttons, a mouse wheel, the keyboard, or menu commands.

Zooming with the Mouse

View > Zoom > Mouse (shortcut **Z**) changes the function of the left mouse button for a single operation to enable a *zoom box*. When the zoom box is enabled, the next two mouse clicks define opposing corners of a rectangle. S-Edit zooms the display window directly to the area inside the rectangle.

You can also click and drag to draw the zoom box. Note that S-Edit must maintain the correct height-to-width ratio of the display, so the new work area may not be exactly the region contained inside the rectangle.

After a **Zoom Mouse** operation the mouse buttons revert to their previous functions.

Zooming with the Mouse Wheel

Spin the mouse wheel up to zoom in and down to zoom out. In both cases the zoom will be centered on the cursor location.

Zooming with the Keyboard

Use the plus key (+) to zoom in by a factor of 1.5 and the minus key (–) to zoom out by a factor of 1.5.

Zooming to Selected Objects

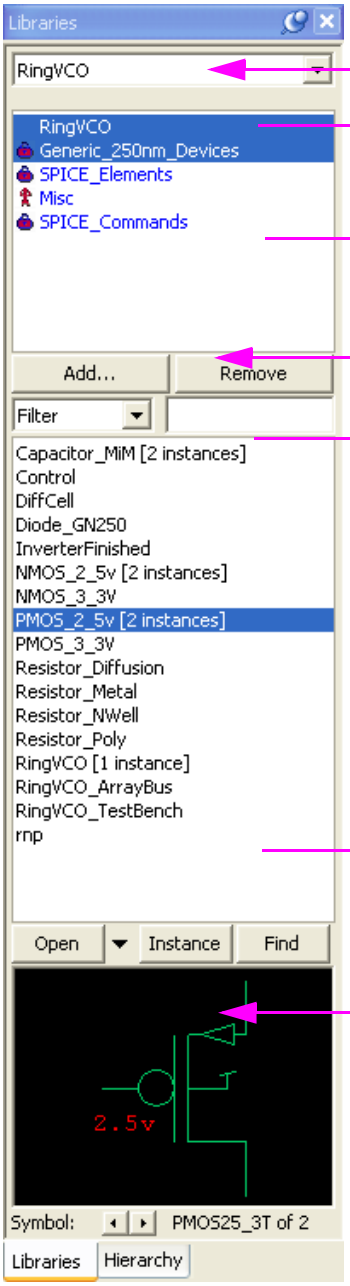
Use **View > Zoom > To Selection** (shortcut **W**) to zoom the display window to encompass only the selected object(s).

Zooming to Show the Entire Contents of a Cell

View > Fit (shortcut **Home**) zooms the display window so that all objects in the design area are visible in the work area.

The Libraries Navigator

The Libraries navigator provides an easy way to view, manage and instance your libraries and cells. You can use it to add and remove cell libraries and to list the cells in all the libraries that are loaded. You can also use this dialog to open, copy, instance or perform several other cell operations (see “Shortcuts for Cell and View Commands” on page 70).



This drop-down menu lets you select from the designs that are currently open.

The **Libraries** navigator shows the designs and libraries that are open.

- (no icon) means a design is open with exclusive access and editing is allowed
- means a design is open with exclusive access and editing is not allowed
- means a design is open with non-exclusive access and editing is allowed
- means a design is open with non-exclusive access and editing is not allowed

The **Cell list** shows the cells available from the design and libraries highlighted in the Libraries navigator.

The **Preview Pane** displays the symbol view for the selected cell. You can right-click anywhere in the Libraries navigator to toggle this preview on or off.

(drop-down menu)

The uppermost field is a drop-down menu that lets you select a design to open from those that are loaded.

Libraries navigator

This pane lists the designs and libraries that are currently open.

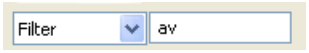
When you highlight a file in the Libraries navigator, the cell browser displays the cells it contains. Use the **Ctrl** key to select multiple libraries and list the cells in each of them. Black text indicates the design file, blue text indicates library files.

Add

Use this button to **Add** libraries. Press **Ctrl** and click to select multiple entries.

Remove Use this button to **Remove** libraries.

Filter



The **Filter** option lets you enter letters or numbers to filter the cell list to show only the cell names that contain those characters. For example, type “av” to show only cells with the letters “av” in their name. The filter is case-sensitive.

The remaining options in the pull-down menu provide hierarchical filters:

- **Top-level**—filters the list to show only those cells that are not instanced in the design.
- **Leaves**—filters the list to show just cells that do not instance other cells (primitive cells).
- **Instanced**—filters the list to show only those cells that are instanced in the current view.
- **Modified**—filters the list to show just those cells that have been changed but not saved.
- **Parents**—filters the list to show only the cells that instance the active cell.
- **Modified Time**—sorts cells to according to the date and time they were last modified, with the most recent at the top of the list.

Cell list Lists the cells contained in each of the files highlighted in the Libraries navigator (and that meet the filter criteria, if any.)

Information in square brackets shows how many instances of a given cell are used in the active view.

Open Use this button to open the highlighted cell. (See also “Opening a View (Cell > Open View or “O”)” on page 71.)

Instance Use this button to quickly instance the highlighted cell. (See also “Creating Instances” on page 115.)

Find Use this button to add all instances of the highlighted cell to the existing selections in the active view.

(preview pane) This preview pane displays the symbol view for the selected cell. You can right-click anywhere in the Libraries navigator to toggle the preview on or off.

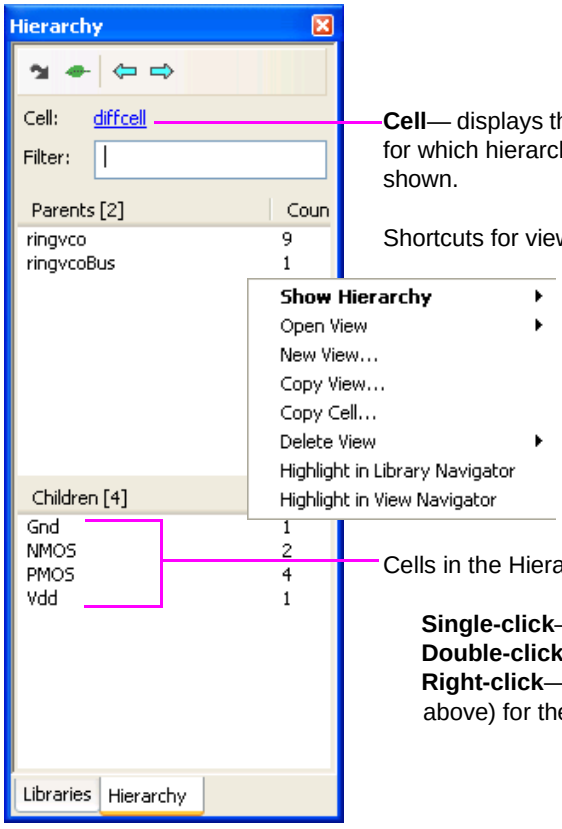
Symbol __ / view_x of y Displays the number, view type and view names for the highlighted cell.

The Hierarchy Navigator

The Hierarchy navigator lets you view the structure of a design by showing a list of *children* and *parents* related to a given "*root*" cell, where **children** are the instances the cell contains, and **parents** are the cells that contain an instance of the root cell.

For any root cell, the Hierarchy navigator provides two views. With *Full hierarchy* disabled, the navigator shows just the immediate hierarchy—one level up to the parents and one level down to the children. When *Full hierarchy* is enabled, the navigator shows the **ancestors**—all levels of instantiation up from the root cell, and **descendants**—all levels of instantiation down from the root cell.

When you change the active view in the design area, the Hierarchy navigator automatically updates to show information for the cell in focus.



Cell— displays the focus (root) cell for which hierarchy information is shown.

Shortcuts for viewing hierarchy

Cells in the Hierarchy list can be used for navigation as follows:

Single-click—highlights the cell name.
Double-click—shifts focus to the highlighted cell.
Right-click—opens the context-sensitive menu (shown above) for the selected cell.

Hierarchy Navigator Toolbar

Full hierarchy— toggles display from all levels of the design hierarchy to just those immediately above and below the focus cell.

Show roots and leaves only— displays only cells at the very top or bottom of the hierarchy—*terminal cells*—with respect to the

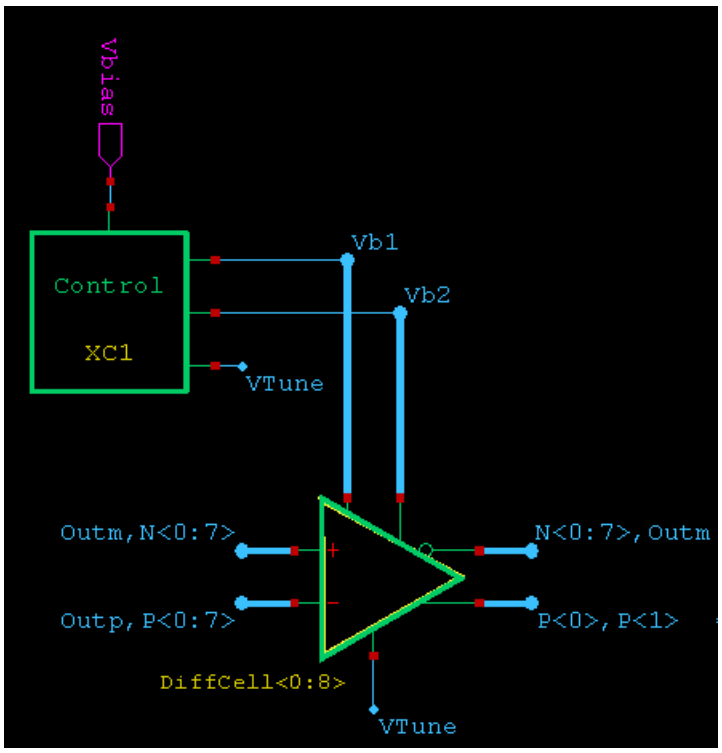
Back— scrolls backwards through the history of root cells viewed in the Hierarchy navigator.

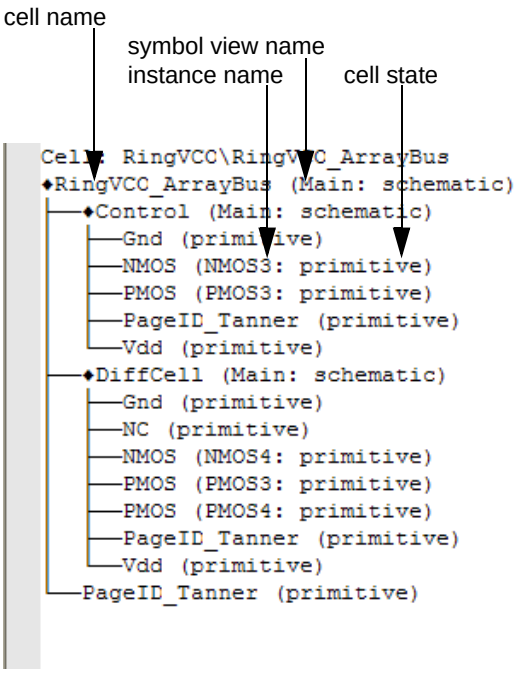
Forward— scrolls forward through the history of root cells viewed in the Hierarchy navigator.

Cell	Displays the name of the root cell for which hierarchy is displayed in the navigator. Note that this field does not always indicate the cell that is in focus in the design area. You can click on this link to change focus in the design area to the cell named.
Filter	Use this text field to filter the navigator lists. It is case- and position-insensitive.
Parents [ ] / [ ] **or** **Ancestors [ ] / [ ]**	**Parents** shows cells one level of hierarchy above it that instance the root cell. **Ancestors** shows cells in all design levels above it that instance the root cell. The value in the first bracket indicates the total number of instances currently displayed in the list. When a second bracket is shown, it indicates the total number of qualifying cells. Click on the column header to sort cells alphabetically.
Count [ ] / [ ]	For the upper list field, shows the number of times a cell is instanced in a cell. For the lower list field, shows the number of each instance in a cell. Click on the column header to sort cells numerically.
Children [ ] / [ ] or **Descendents [ ] / [ ]**	**Children** shows cells one level of hierarchy below it that instance the root cell. **Descendants** shows cells in all design levels below it that instance the root cell. The value in the first bracket indicates the total number of instances currently displayed in the list. When a second bracket is shown, it indicates the total number of qualifying cells. Click on the column header to sort cells alphabetically.

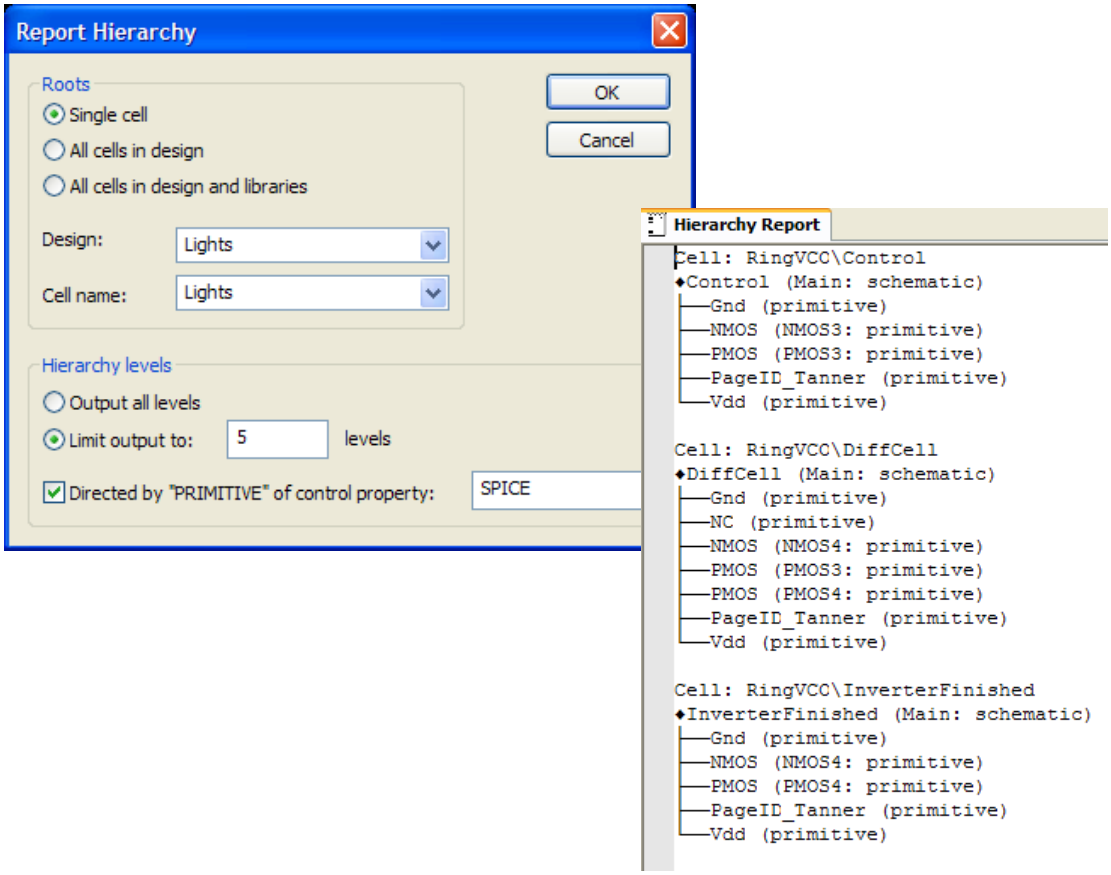
Hierarchy Report in Text Format

You can also generate a text file that shows the hierarchical contents of a cell, design, or design and its libraries.





Use **File > Hierarchy Report** to open the **Report Hierarchy** dialog.



Roots — Use these buttons to select the extent of the report from **Single cell**, **All cells in design** or **All cells in design and libraries**. There are pulldown fields to select the **Design** and the **Cell name**.

Hierarchy levels — You can choose to either **Output all levels**, or you can enter an integer value which will set the number of levels, in the **Limit output to** field.

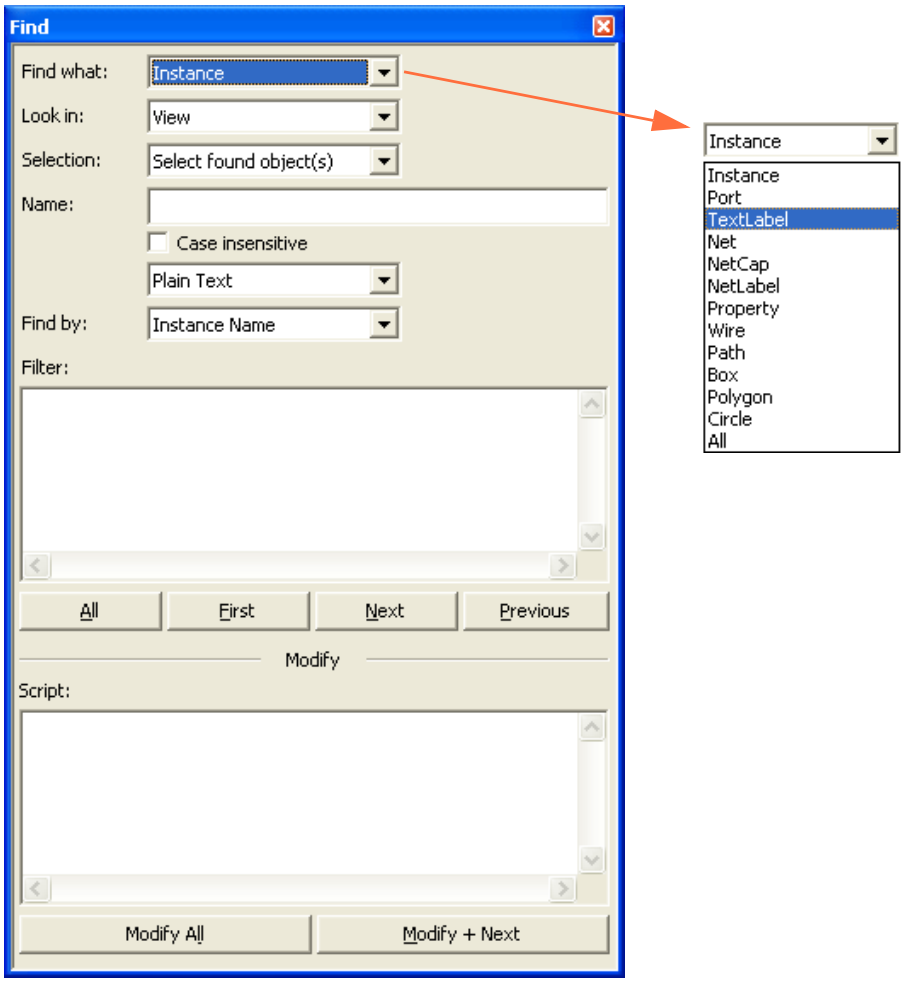
Directed by the "PRIMITIVE" of control property: — If you enter a value in this field, the hierarchy tree of that value will not be displayed in the report.

The Find Navigator

The Find navigator is a powerful search tool that allows you to set your search over one or more designs, by object, by object name and, using commands, even specific parameter values. For example, with a script `string match P* [ property get -system Name ]` you can find all instances whose name starts with "P". S-Edit supports case-insensitive searches, and you can also search using regular expressions or wildcards in addition to explicit values.

Note that the default behavior is to find all objects that meet the criteria. Also, in searches for NetLabel, NetCap, Port and Instance object types, the default search (as you can see in the Command Line

window) is “-exact.” To treat names as bundle names, for example if you want to find **Vb1<0:7>**, you would use the qualifiers “-and” and “-or.”



Find What

Use this pull-down to select the object type to search for, such as instance, port, text label, net, etc. **All** selects all types of object that fit the search criteria.

Look in

Select the scope of the search.

- **Selection**—search only within the objects that are selected in the active view.
- **View**—search within the active view.
- **Design**—search the entire active design.
- **Hierarchy**—search down the hierarchy of the active cell.

(continued)

Selection

Sets the selection behavior when objects matching the search criteria are found.

- **Select Found Object(s)**—selects only objects that match the search criteria.
- **Add to Selection**—objects that are currently selected remain selected and objects that match the search criteria currently displayed in the **Find** navigator are also placed in the selected state.
- **Remove from selection**—objects that are currently selected remain selected unless they match the search criteria currently displayed in the **Find** navigator, in which case they are deselected.

Name

Enter the name(s) of the object to search for, and use the search mode field (see below) to select how the **Name** entry is matched.

Case insensitive

Check this box to make your search case insensitive.

(search mode)

Select the type of search to perform. Note that this field must be set to properly match the value(s) in the **Name** field. For example, the value "***Vb****" in the Name field will only function as a wildcard search when this field is set to "**Wildcards**."

- **Plain Text**—use this to search for the text in the **Name** field.
- **List of Plain Texts**—use this to search for a list of objects entered in the **Name** field, separated by a comma or semicolon.
- **Wildcards**—use this to interpret ***** or **?** in the **Name** field as a wildcard character.
- **Regular expressions**—matches objects that meet criteria entered in the form of a PERL regular expression in the **Name** field.

Net

Only when the search object is set to **Net**, this field applies a filter to the **Name** field.

- **Exact**—filters the search criteria to find a unique net.
- **And**—filters the search criteria to find all nets of a name or range, such as those that originate from a bus or bundle.
- **Or**—filters the search criteria to find any of the net names entered.

(continued)

Filter

Use this field to enter TCL scripts that filter a search. **Filter** scripts are particularly useful for selecting instances based on their parameter values.

Note that TCL script in this field is evaluated in the context of just one object being selected. That is, TCL commands entered here are applied only to objects that meet the criteria set in the fields above it, and such objects are evaluated individually. This is important to keep in mind—your script may include multiple criteria, but those criteria are applied to only one object at a time.

If the script returns false or 0, then the object is discarded from further operations. If the script returns true or a nonzero value, then the object is successfully found and added to the list of selected objects.

For example, the script below selects transistors that have gate lengths within a specific range of .25e-6 and .35e-6. Note that it will generate a warning message for instances which do not have a length property.

```
set L [ property get L -double -nowarn ]
expr { 0.25e-6 <= $L && $L <= 0.35e-6 }
```

This example selects all instances with a local value for L:

```
if { ! [ property get L -exists ] } { return false }
  return [ property get L -islocal ]
```

All

Use this button to initiate a search. Searches for and selects all objects at once.

First, Next, Previous

Use these buttons to search for and select objects one at a time.

These three buttons operate in the context of the active view. Each time the **First** button is pressed, the view context is reset. For this reason you should use **First** to initiate a search whenever you plan to view the results individually or sequentially.

First—highlights the first object that matches the search criteria, where the ordinal “first” is established internally according to Tanner database order.

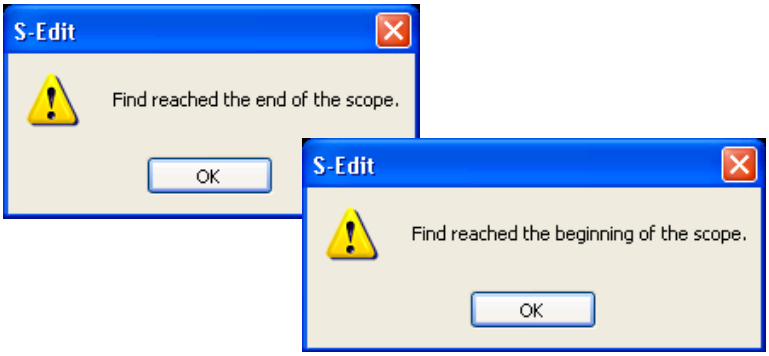
Next—selects the next object that meets the current search criteria. This command moves successively through each view containing one or more objects that match the search criteria until reaching the last object. However if the “first’ object is reset, the scope and order of objects meeting the search criteria will also reset.

Previous—selects the previous object that meets the current search criteria, with behavior matching that of the **Next** button.

(continued)

First, Next, Previous (continued)

S-Edit will indicate when you have reached either the first or last object that matches the search criteria as shown below.



Modify Script

Use this script field to modify objects that meet the preceding criteria:

TCL script in this field is applied to the individual elements in the **Find** list. The script is applied once for each element, and is applied in the context of one element selected at a time. For example, the script below moves each selected instance up by ten database units:

set y [property get Y -system]
property set Y -system -value [expr { $y + 10 }] -units iu

Modify All

Applies the script in **Modify Script** to all objects. To revoke modifications made with this command, you will need to use the **Undo** operation in each of the views containing an object that was modified.

Modify & Next

Applies the script in **Modify Script** to the currently selected found object and navigates to the next one. Modifications made with this commands can be reversed using **Undo**.

Finding All Instances with a Certain Property

To select all instances that contain a specific property name, search for Instances in the **Find What** field, keep the **Name** field blank, and in the **Filter** field enter an expression that matches the desired combination of property and value.

For example, *`expr [property get Delay] == 4`* will select all instances with a property “Delay” of value “4”. This type of search is helpful if, for example, you have several per instance properties that were added but that are not in the symbol view for a cell.

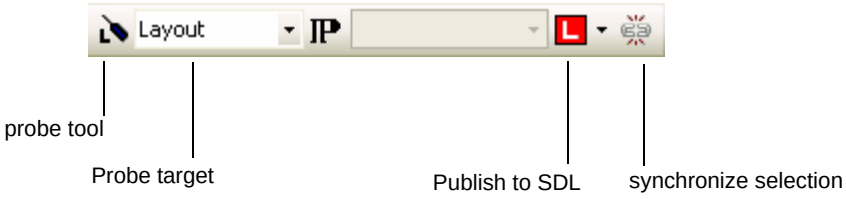
Cross Probing from Schematic to Layout

You can cross probe from an instance or net in schematic to the corresponding instance or net in layout, and vice-versa.

Note that design hierarchy is importantand must match. For cross probing to work, the device must have the exact same name in layout as in schematic, and the corresponding design must be open in L-Edit.

Probing and Selection Synchronization

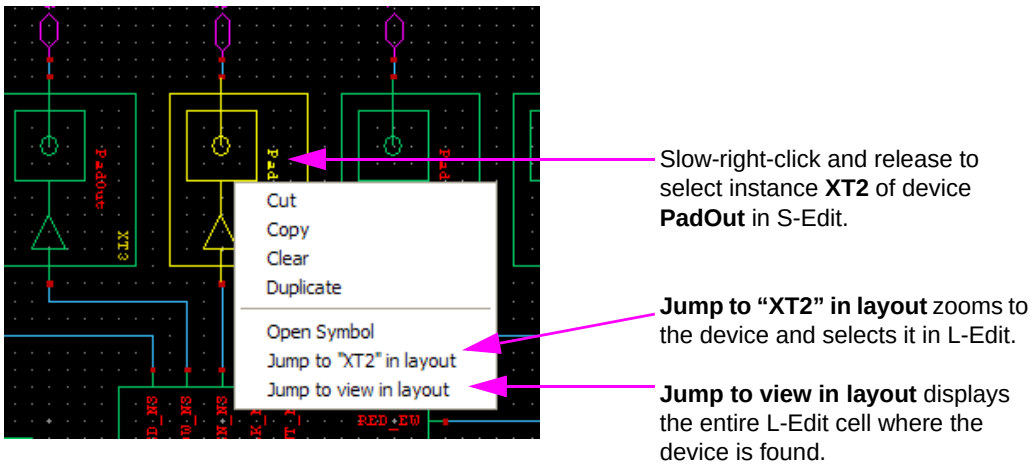
S-Edit has two modes of cross probing, i) using the probe tool, and ii) using selection synchronization. To use the probe tool, select the probe target as Layout, then press the **Probe to layout** toolbar button. To use Selection Synchronization, press the **Synchronize Layout Selection** button. Using the probe tool you can click on instances or nets in schematic, and the corresponding instance or net will be highlighted in layout, and L Edit will pop to the front. With Synchronize Selection, selecting an instance in net in schematic using the normal selection methods will select the corresponding instance or net in layout, but L-Edit will not pop to the front.



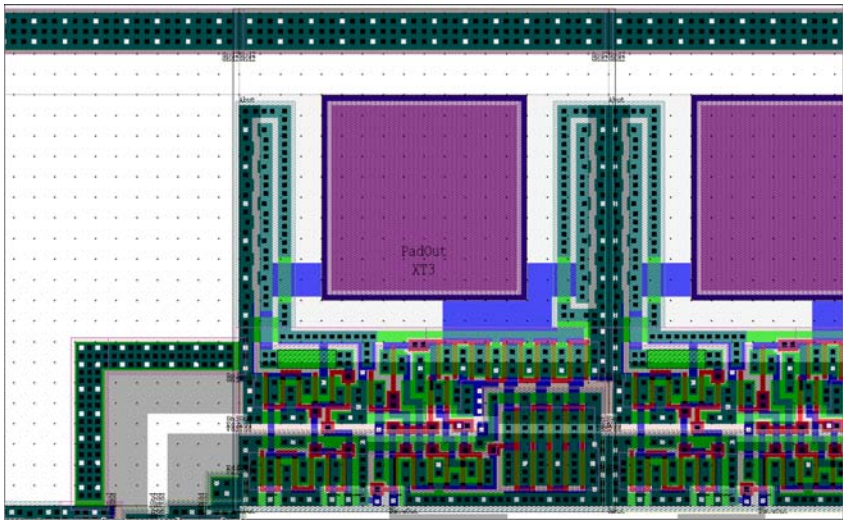
Jumping to a Device in Layout

You can also cross probe to a device from schematic, using the commands **Tools > Jump > Jump to device in layout**, **Jump to view in layout** or **Jump to net in layout**.

You can also "*slow-right-click*," that is, press and hold the right mouse button over the desired object. When you release the mouse button, S-Edit displays a menu that includes the options to jump to the device, view or net, as appropriate, in layout.


Cut
Copy
Clear
Duplicate
Open Symbol
Jump to "XT2" in layout
Jump to view in layout
Slow-right-click and release to select instance **XT2** of device **PadOut** in S-Edit.
Jump to "XT2" in layout zooms to the device and selects it in L-Edit.
Jump to view in layout displays the entire L-Edit cell where the device is found.

Once you initiate the jump, the L-Edit button in the taskbar will flash.

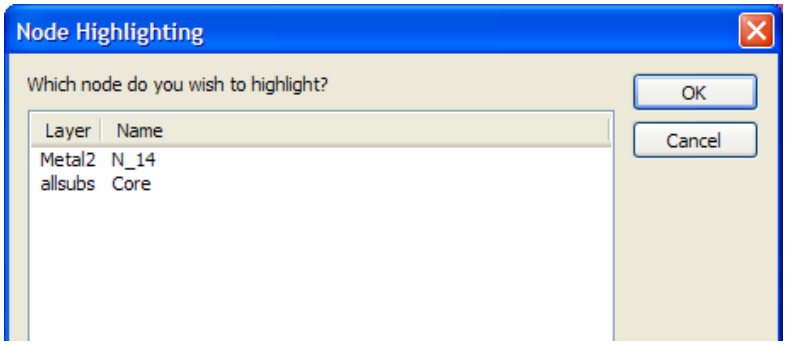


When you select “Jump to “XT2” in layout in S-Edit, the L-Edit window zooms in and centers on instance XT2 of cell PadOut.

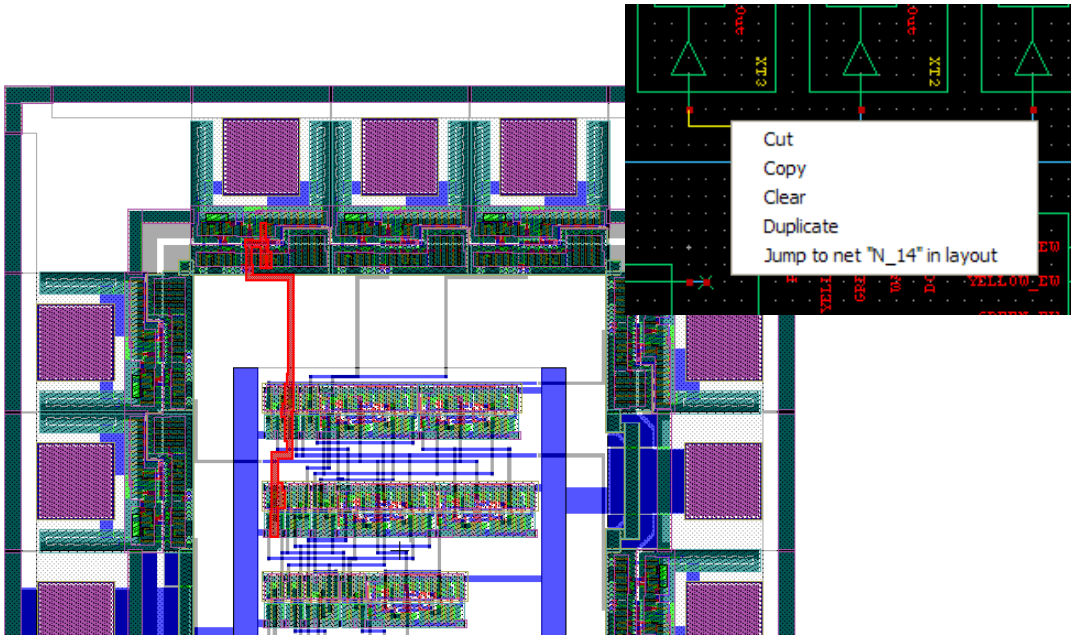
Selection: Instance 'XT3' of cell 'PadOut'

Jumping to a Net in Layout

When you jump to a net from S-Edit, L-Edit uses the node highlighting feature to search for a node name—so the corresponding net in L-Edit must contain a port having the same name as the net in S-Edit. You may also need to pick the net you are interested in from those the port (node) belongs to.



L-Edit will then highlight the corresponding net.



4 Drawing, Selecting and Editing Objects

Object Types

Drawn objects in S-Edit are either *electrical objects*—for example, wires, ports or solder points, or *annotation graphics*—geometric objects without electrical significance.

To draw an object, you must select a tool from either the *Drawing toolbar* (see “Drawing Tools for Annotation Graphics” on page 100) or the *Electrical toolbar*, (see “Drawing Tools for Electrical Objects” on page 104) and also a *segment type* from the *Segment toolbar*.

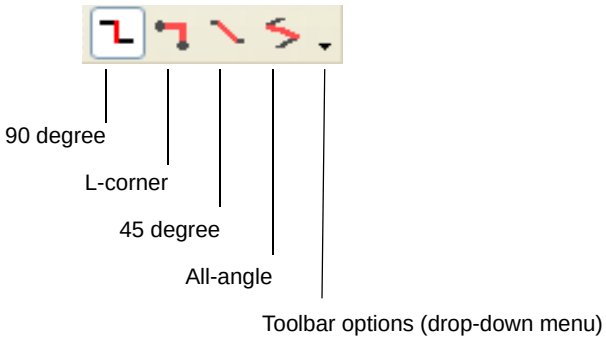
Annotation and electrical tools are “sticky”—once you select a tool, S-Edit will draw only objects of that type until you choose a different tool.

You can change segment type at any point while drawing a geometric object, for example from orthogonal to 45° segments.

Segment Types

S-Edit has various types of lines segments—90°, L-corner, 45° and all-angle—you can select from when drawing. You can change segment type at any point during a drawing operation by simply selecting a different icon from the Segment toolbar.

To draw a line, select **Path** and a segment type, then position the cursor where you want to place the starting point. **Left-click** to begin drawing and to place vertices. **Double-click** to place a vertex and end drawing. **Right-click** to back up to the vertex that was placed last.



90° Segments

When you use this segment option, S-Edit will only draw geometry having 90° angles.

L-Corner Segments

The L-corner segment draws an L-shaped line from the last placed vertex to the current cursor position.

You can toggle the orientation of the 90° angle by pressing the “L” key before placing the next L-shaped segment.

Note that the L-shaped line behavior remains active when you use the middle button, so the action of removing vertices is modified to remove two vertices with each middle-button click.

You can also invoke this mode with the following command line commands:

mode -drawstyle hv, where the horizontal segment is first, and **mode -drawstyle vh**, where the vertical segment is first.

45° Segments

When you use this segment option, S-Edit will draw geometry having 45° or 90° angles.

All Angle Segments

With this segment option, S-Edit will draw geometry having any angle.

Using the Mouse to Draw

Drawing instructions are given for the most commonly-used drawing interface, a three-button mouse (with or without a mouse wheel), as default. Other drawing methods are discussed in “Command Window” on page 28 and “Drawn Properties of Objects” on page 105.

Use the mouse as follows for drawing operations:

LEFT-CLICK

Use a left-click (**DRAW**) to start a line segment or place a vertex.

MIDDLE-CLICK

Use a middle-click (**BACKUP**) to remove vertices in reverse order of placement. This back-up action is only available while an object is being drawn. The **BACKUP** button can also be used to move a vertex.

DOUBLE-CLICK

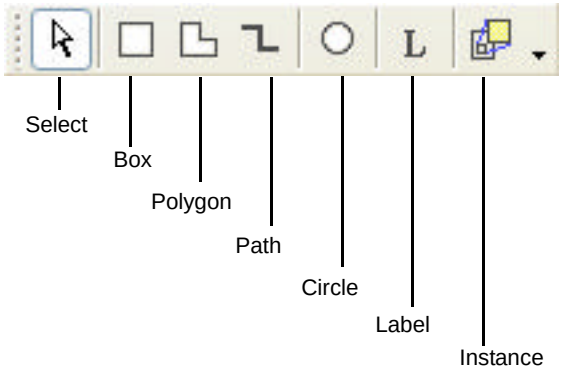
Use a double-click (**END**) to end the drawing operation by placing a vertex.

RIGHT-CLICK

Use a right-click to end the drawing operation without placing a vertex. (This results in what can appear to be "backing up" to the last vertex placed.)

Drawing Tools for Annotation Graphics

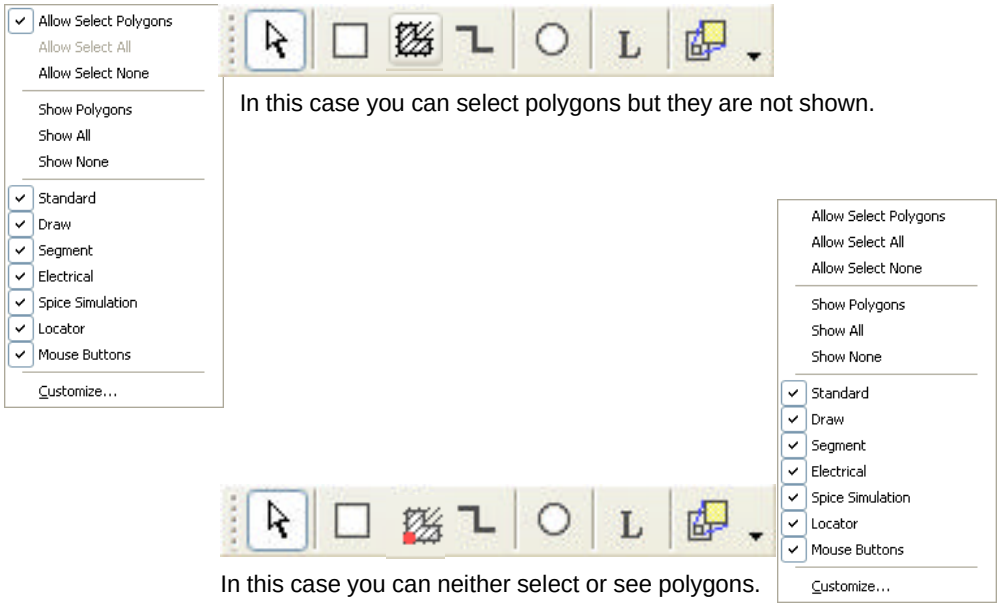
Annotation tools are available in any view type.



You can allow or disallow selection of the objects shown above, and also of net labels, ports and properties, by type. Similarly, you can show or hide objects based on type.

Right click on the toolbar button for an object type to choose selectability and visibility. Note that though labeled “Show None,” in fact at least one type of object must be displayed at any given time; it will be the object type over which the cursor was right-clicked to open the context-sensitive menu.

Selectability and visibility of an object type is indicated in the toolbar buttons. Diagonal hatching means the object type is not visible. A red dot in the lower left corner of the button means the object type cannot be selected.



Boxes (Draw > Box or "B")

To draw a box, press the DRAW mouse button and drag the cursor away from the starting point to determine the opposite corner (and therefore the length and width) of the box. Release the DRAW button at the desired opposite corner.

Polygons (Draw > Polygon or "P")

To draw a polygon, click the VERTEX (left) mouse button at the starting point, move the cursor and LEFT CLICK to determine the second vertex. Repeat the process for each successive vertex. Click the BACKUP mouse button to remove the last vertex that was placed. A polygon can have any number of vertices.

When you click the END button, coincident vertices (two or more vertices occupying the same location) and colinear vertices (three or more vertices lying on the same straight line) are eliminated.

Unlike L-Edit, S-Edit uses incremental rendering so that when you are drawing a polygon it only closes when you complete the drawing.

When you draw a polygon the mouse buttons become VERTEX, BACKUP, and END, respectively.

Paths (Draw > Path)

Note that this is simply a line drawing tool—a path has no electrical properties. A path can have any number of vertices.

To draw a path, click the VERTEX (left) mouse button at the starting point, move the cursor and LEFT CLICK to determine the second vertex. Repeat the process for each successive vertex. Click the BACKUP mouse button to remove the last vertex that was placed. Click the END button to complete the operation.

The mouse buttons for this operation are VERTEX, BACKUP, and END.

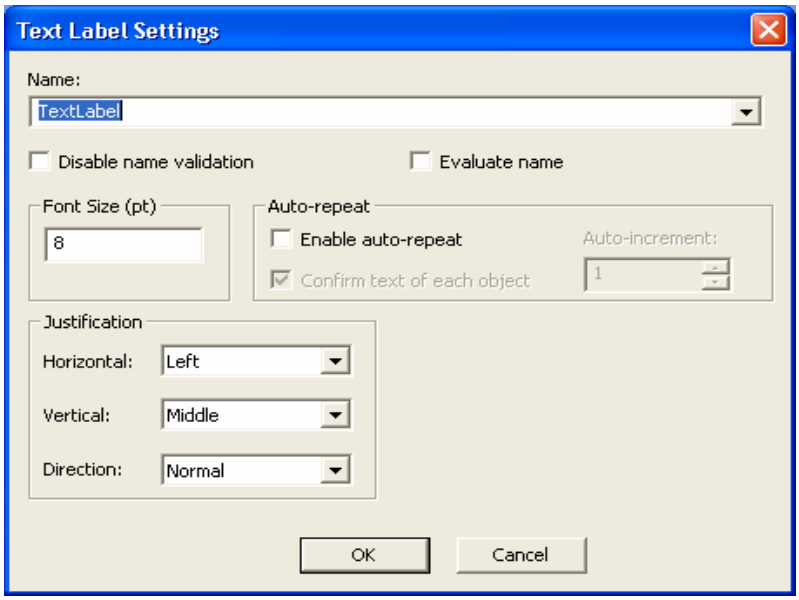
Circles (Draw > Circle)

To draw a circle, press the DRAW mouse button and drag the cursor away from the center starting point to determine the radius of the circle. Release the DRAW button at the desired radius.

The mouse buttons for this operation are DRAW and SELECT.

Labels (Draw > Label)

When you select the **Label** icon and click in the design area, S-Edit opens the **Label Settings** dialog so you can enter text and choose how it will be displayed. Click **OK** to place the label.



Name Enter your label text in this field.

Font Size Enter the text size.

Evaluate name If **Evaluate name** is checked then the label will be evaluated when S-Edit is in Display Evaluated mode.

Auto-repeat If **Enable auto-repeat** is not checked, S-Edit opens the **Label Settings** dialog just once, places the label when you click **OK**, and reverts to **Select** mode.

With **Enable auto-repeat** checked you can place labels repeatedly by clicking in the design area, without having to re-open the label settings dialog each time. S-Edit remains in label mode until you select a different mode or press **ESC**. The following controls may be added:

- **Confirm text of each object**—when checked, re-opens the **Label Settings** dialog each time a label is placed so you can enter changes to the next label placed.
- **Auto-increment**—when **auto-repeat** is active and the last character in the **Text** field is an integer, S-Edit automatically increments the label text by the value entered here.

Justification

All text orientations are with respect to the origin point of the label, indicated by a cross.

Horizontal: select **Left**, **Center** or **Right** horizontal text alignment.

Vertical: select **Top**, **Middle** or **Bottom** vertical text alignment.

Direction: select **Normal**, **Upside down**, **Down** or **Up** for the text direction.

Auto-Repeat for All Types of Labels

For labels, port labels and net labels in S-Edit, when **Enable Auto-repeat** is checked, you will remain in label mode so you can place multiple consecutive labels. Otherwise, S-Edit opens the label dialog just once, places the label when you click OK, and reverts to Select mode.

If **Confirm text of each object** is checked, the label dialog will reopen with each placement, allowing you to change the any setting. If it is not checked, the dialog will not reopen, and you can place as many identical consecutive ports as desired.

If the label name ends in a numeral and **auto-repeat** is active, when you place labels in succession, the number will increment by the amount entered in the **Auto-increment** field.

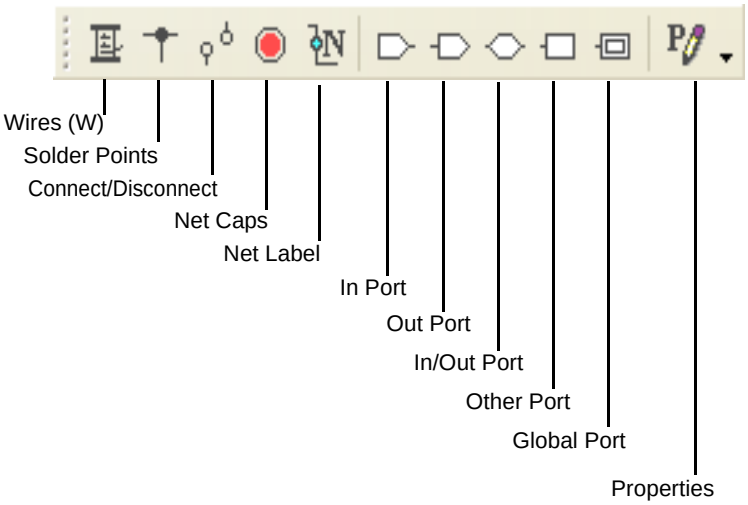
Note that you can use the **R**, **H** and **V** shortcut keys (for rotate 90°, flip horizontally and flip vertically) to change the orientation of a label before clicking to place it in the design area. This, as opposed to using the Properties navigator, is the suggested procedure.

Instance (Cell > Instance)

To instance a cell, highlight it in the Libraries navigator and click on the **Instance** icon (for more details, see “Creating Instances” on page 115).

Drawing Tools for Electrical Objects

In S-Edit, design connectivity is achieved by the proper arrangement of ports, labels, and wires to form nodes. They are drawn using the **Draw > Electrical** menu or with the electrical toolbar (only available in schematic view).



Wires (Draw > Electrical > Wire)

Note that wires and paths, while similar in appearance, are functionally different. Wires connect objects electrically; paths do not. S-Edit will draw only orthogonal wires.

The starting point is the first vertex of the wire. Wires can have any number of vertices. The mouse buttons for this operation are VERTEX, BACKUP, and END. See “Drawing Wires” on page 120.

Solder Points (Draw > Electrical > Solder Point)

Use solder points to define a point where wires cross and make an electrical connection (see “Creating a Connection where Wires Intersect” on page 121).

Connect/Disconnect (Draw > Electrical > Connect)

The Connect/Disconnect tool attaches and detaches wires within its perimeter. (See “Rubberbanding and Disconnecting Wires” on page 122.)

Net Caps (Draw > Electrical > Net Cap)

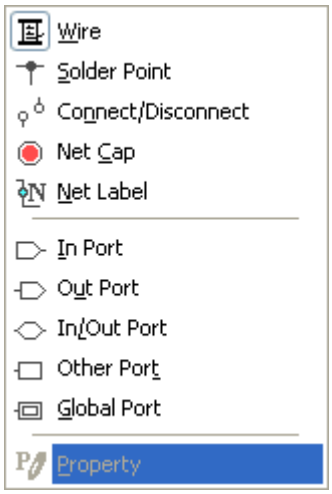
A net cap halts the propagation of a global net to any cell that instances the current cell. (See “Naming Global Nets” on page 133.)

Net Labels (Draw > Electrical > Net Name)

Net labels identify nets by placing text in the design area. Use the **NetLabel Port Settings** dialog to place text and control its size, position and alignment. (See “Labeling Nets” on page 125).

Ports

Use the toolbar icons or the **Draw > Electrical** menu to select and place ports. S-Edit will open a dialog where you can set the port name, size and alignment parameters. (See “Adding Ports” on page 122.)

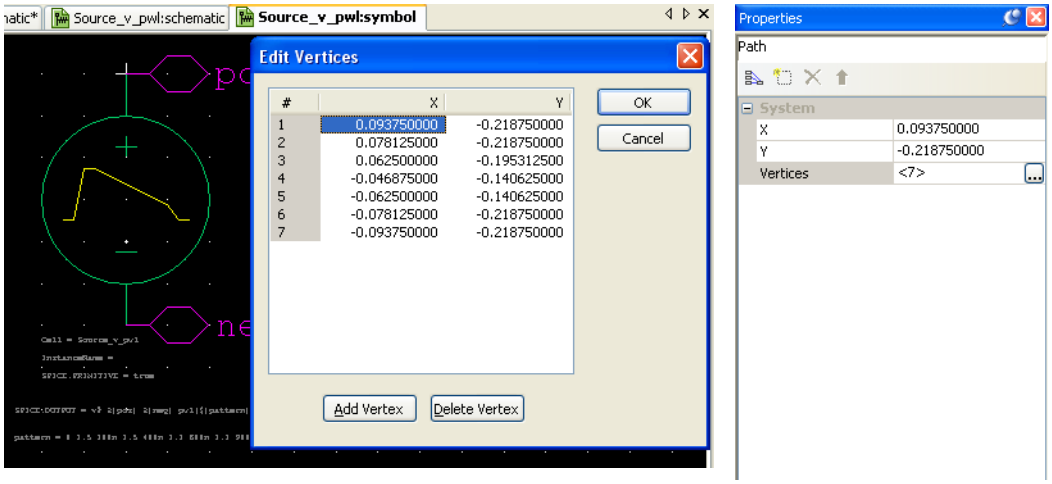


Drawn Properties of Objects

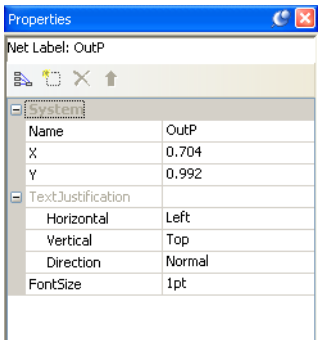
Cells and objects in S-Edit are characterized by drawn properties such as the coordinates of their vertices, the length and width of their lines, etc. You can use these properties to edit objects by changing the related values.

The set of drawn properties differs for each object. For example, a path, polygon, or wire is characterized by its vertex coordinates (**X1**, **Y1**), (**X2**, **Y2**), (**X3**, **Y3**), etc.

Note: To open a dialog like the one below, click once in the **Vertices** field of the Properties navigator to activate the ellipsis button and then again on the button to open the **Edit Vertices** dialog.



Similarly, a comment or net label is characterized by its name, location in coordinates, text size and orientation.



Electrical Properties of Objects

See “Symbol Property Types” on page 147 and “Editing Instance Properties” on page 117 for information on electrical properties.

Selecting Objects

You must select an object before editing it or modifying it in any way. You can select more than one object at a time to manipulate a group of objects as a single one.

When you select an object, it is highlighted. Or, if the object is an instance of a cell, a thin highlight appears defining its MBB [minimum bounding box, or perimeter].

If the **Display** property under **Name** is set to **Visible** or **ValueOnly**, the name will also be displayed in with yellow highlighted text when a cell or instance is selected. (If the names are not visible try zooming-in, or set the display to a larger **FontSize** value.)

When you deselect an object, the highlight disappears and subsequent operations do not affect that object.

S-Edit provides two ways of selecting objects: explicit and implicit selection.

Explicit Selection

There are several ways to make an explicit selection. You must select objects explicitly before performing the following actions:

- **Cut**, **Copy**, **Clear**, or **Duplicate** operations.
- Operations on a set of objects.
- Operations on one of several close or overlapping objects.

Selection by Clicking

Position the cursor over the desired object and click the **Select** button. S-Edit will deselect any previously selected object or objects.

Selection by Enclosing (Selection Box or type Select - Enclose)

Drag the cursor with the **Select** button held, forming a *selection box* around the objects. Draw the selection box so that it completely encloses all the objects you want to select but no others. (To select a port, you must enclose the starting point, but not the port text.) S-Edit will deselect any previously selected objects. (See “Mouse Wheel Options” on page 45.)

Selection by Intersection (type Select - Intersect)

When you type select-intersect in the Command window, objects that are intersected by the selection box (i.e. partially enclosed) will be selected as well as fully enclosed objects. (See “Mouse Wheel Options” on page 45.)

Extend Selection

To add an object to a set of selected objects, perform a click- or drag-selection with the Extend Select (**Shift + Select**) button held. Previously selected objects are *not* deselected.

Cycle Selection

When you click repeatedly with the pointer in the same spot, L-Edit uses “*cycle selection*” to successively select each object within a defined distance from the cursor location. This option lets you move quickly between overlapping or adjacent objects to select the one of your choice.

Which objects are selected is determined by the selection range you have set for implicit selection (see “Selection Behavior Options” on page 46.)

The order of selection is determined by the distance from the point clicked to the nearest boundary of each object. The first click selects the nearest object, the next click moves selection to the next closest object and so on within the selection range.

Selection proceeds regardless of object type. Once each object within the selection range has been selected, the next click deselects all objects and the following click restarts the cycle beginning with the closest object. If you click inside a single object, that object is selected first. If you click inside overlapped objects, the object with the nearest interior boundary is selected first and selection proceeds through the next nearest of the overlapped objects and then outside them, from nearest to furthest object.

Select All (Edit > Select All) and Deselect All (Edit > Deselect All)

Use **Ctrl + A** to select all objects in the active window and **Alt + A** to deselect all objects in the active window.

To remove objects from a group of selections, hold the **Deselect** (**Alt + Select**) button while performing a click- or drag-selection.

Limiting Selection by Type of Object

You can allow or disallow selection of boxes, polygons, paths, circles, wires, labels, netlabels, ports and properties by object type.

Right-click on the toolbar button for an object type to open a context-sensitive and click on the “Allow Select...” option you prefer. A red dot in the lower left corner of the toolbar button indicates an object type cannot be selected.

Implicit Selection

You can **move**, **edit**, or **copy** an object without explicitly selecting it, if it is within the selection range set in “Selection Behavior Options” on page 46.

Clicking the MOVE-EDIT or COPY button with the cursor near such an object implicitly selects it for these operations. The object is deselected after the operation.

Deselection

To deselect an object or remove objects from a set of selected objects, hold the **Deselect** (**Alt + Select**) button while performing a click- or drag-selection.

Deselecting an object that was selected has no effect on the object.

Automatic Deselection

When you begin a move, edit, or copy operation, S-Edit checks to see how far the cursor is from the bounding box of all currently selected objects. If this distance is greater than the deselection range, then S-Edit deselects all current selections and performs the implicit selection to begin the operation.

Moving and Editing Objects

To move an object, select it (being careful not to select just an edge or vertex) and drag the object to the new position while holding the MOVE-EDIT button. To move multiple objects you must explicitly select them. S-Edit surrounds the objects with a *selection box* and maintains their relative positions during the move.

The function of the MOVE-EDIT button will change depending on the position of the cursor. If you click on or very near an object's edge or vertex with this button, you will move only that edge or vertex and thus change the object's size or shape.

Use Mouse Wheel Options to specify the distance from a vertex or edge at which S-Edit will perform an edit rather than a move.

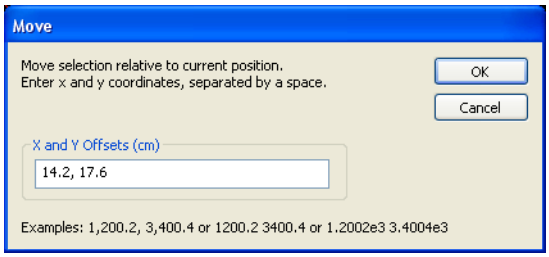
Maintaining Connectivity while Objects are Moved—"Rubberbanding"

Wires in S-Edit "rubberband"—that is, when you move an object, the wires remain connected (unless the force move operation is used. See "Forcing a Move Operation Instead of an Edit (Draw > Force Move or Alt + M)" on page 109.)

Moving Operations

Moving by a Specific Amount (Draw > Move By)

Use **Draw > Move By** to enter an amount by which the selected objects will be moved in the x and y direction.


Move
Move selection relative to current position.
Enter x and y coordinates, separated by a space.
OK
Cancel
X and Y Offsets (cm)
14.2, 17.6
Examples: 1,200.2, 3,400.4 or 1200.2 3400.4 or 1.2002e3 3.4004e3

Forcing a Move Operation Instead of an Edit (Draw > Force Move or Alt + M)

In normal drawing mode, the **edit** command is active when the cursor is within the edit range of an object's edge (see "Mouse Wheel Options" on page 45), and the **move** command is active when the cursor is outside that range.

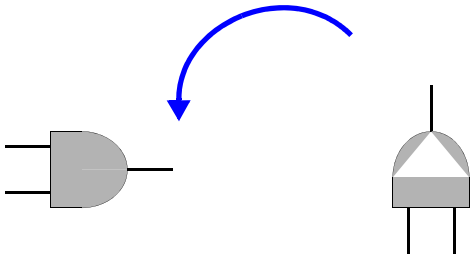
Draw > Force Move forces a move operation regardless of the cursor position. After the move operation is finished, S-Edit will revert to normal drawing mode, or you can cancel the force move operation by pressing **ESC**.

Note that a forced move does *not* preserve an object's electrical connections; that is, it does not rubberband the object.

Rotating Objects (Draw > Rotate 90 degrees or R)

Draw > Rotate 90 degrees (shortcut key **R**) rotates the selected objects by 90° counterclockwise about their starting point. For a single object the center of rotation is the object's origin. For multiple objects the center of rotation is the center of their MBB.

Note that you can use the **R** shortcut key to change an object's orientation before clicking to place it in the design area.

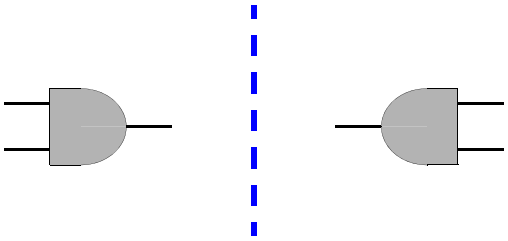


Alternately, the **Draw > Rotate** command allows you to rotate the selected objects counterclockwise by any degree around the starting point, with up to six decimal points accuracy. Use a negative value for clockwise rotation.

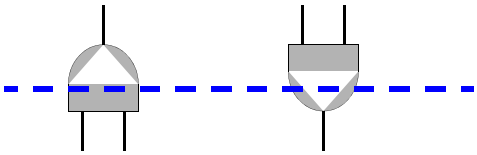
Flipping Objects (Draw > Flip)

These commands flip the selected objects through the center of their MBB, either horizontally (**Draw > Flip > Horizontal**, shortcut key **H**) or vertically (**Draw > Flip > Vertical**, shortcut key **V**).

Note that you can use the **H** and **V** shortcut keys to change the orientation of an object before you click to place it in the design area.



Draw > Flip > Horizontal (shortcut key **H)**



Draw > Flip > Vertical (shortcut key **V)**

Copying and Duplicating Objects

To copy objects, select the object(s) and do one of the following:

- Use the **Edit > Copy** (**Ctrl + C**) command and then the **Edit > Paste** (**Ctrl + V**) command.

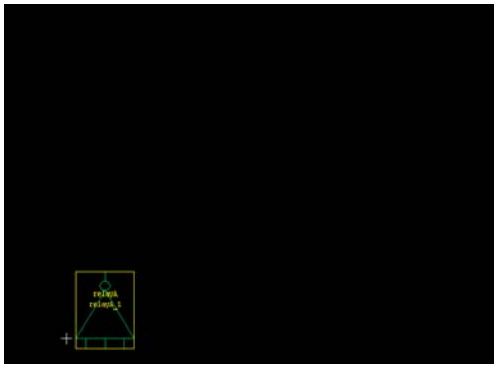
- Use the **Edit > Duplicate** (**Ctrl + D**) command.

The **Copy** command puts the copy of the selected object(s) in the internal clipboard. The copy does not appear in the design area until it is placed with the **Paste** command. The copy remains selected after this operation.

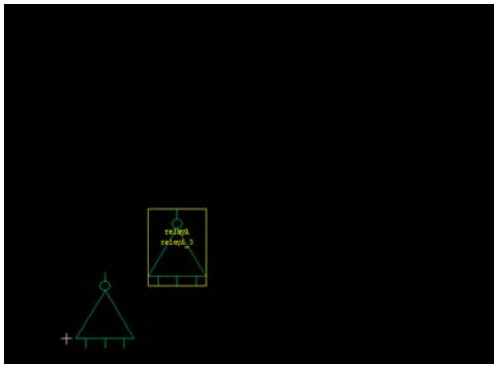
Creating Arrays Using the Duplicate Command

The **Duplicate** operation creates a duplicate of the selected object(s) and places it in the active cell, offset from the original by one grid point horizontally and vertically. The new objects stay selected and can be moved to a new offset.

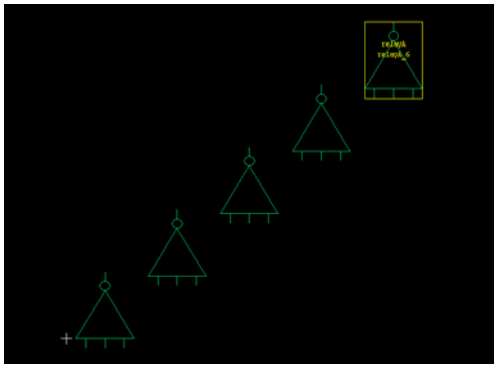
When you use the mouse-drag or **Duplicate** operation, S-Edit stores the offset of that operation. Subsequent use of the **Duplicate** command replicates both the object and the displacement, aiding in the rapid creation of regular structures like arrays. (See “Creating an Array” on page 129.)



Select an object and use **Edit > Duplicate** (**Ctrl + D**) to initiate the duplicate operation.



Click to place the first copy and set the offset.



Use **Ctrl + D** to continue to place copies with the same offset.

Pasting Objects

S-Edit stores cut or copied objects to an internal clipboard. To paste the contents of this clipboard, click in the design area, use the **Edit > Paste** (**Ctrl + V**) command, and click again.

When the paste command is active the clipboard objects are visible and move with the cursor (attached at the lower left corner of their MBB) until the second click places them.

Pasting Objects to Other Applications

You can use **Edit > Capture Window** in an S-Edit design file and to copy and paste the visible design window to another application in bitmap format.

Using Undo & Redo

Edit > Undo (Ctrl + Z)

You can use the **Edit > Undo** command (**Ctrl** + **Z**) to reverse the most recent edit operation and restore the S-Edit file to the state it was in before that operation. Specifically, **Undo** reverses a move; any resize, reorient or reshape operation; the draw, copy, or duplicate operations; deletions, and changes made using **Edit > Edit Object**.

Edit > Redo (Ctrl + Y)

Similarly, you can use the **Edit > Redo** command (**Ctrl** + **Y**) to restore changes reversed with a previous **Edit > Undo** operation.

Showing and Hiding Objects by Type

You can show or hide objects in the layout by object type.

Right-click on the toolbar button for an object to set visibility for that object type. Note that though labeled “Show None,” in fact at least one object type must be displayed at any given time; it will be the object type over which the cursor was right-clicked to open the context-sensitive menu.

Diagonal hatching on the toolbar button indicates an object type is not visible.

Deleting Objects

Use one of the following commands to remove selected object(s) from a view:

- **Edit > Cut** (**Ctrl + X**)

The **Cut** command puts the deleted objects in the internal clipboard. From there they can be restored to the current cell or placed into another cell.

- **Edit > Clear** (**Del**)

The **Clear** command does not put the deleted objects into the internal clipboard.

5 Creating a Schematic

A schematic for a component is made by placing a number of symbols on the schematic page and then connecting the symbols together with wires at their defined connection points. A set of connection points connected together is called a net. You can make a symbol for a new schematic and then use it in other schematics, thus allowing for the construction of hierarchical designs.

Elements of a Schematic View

A schematic view contains any of the following elements:

Instances of Symbols

Instances of symbols refer to a particular symbol in a cell. A schematic may contain many instances of the same symbol or of different symbols. Instances contain graphics which provide the illustration of the symbol, and ports

Ports

Ports provide connection points for attaching nets. Ports on the symbol and ports on the instance are different in that ports on the symbol are shown as their port name, whereas ports on the instance do not. Ports on the schematic define how connections made to ports on symbol instances connect to nets on the schematic for that symbol.

Nets

A net is a wiring connection between two or more instance ports. A net can be a single wire, or a collection of wires called a bus or a bundle.

Properties

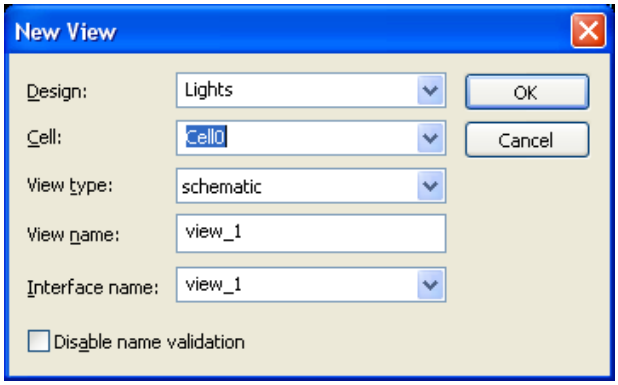
Properties are name-value pairs that are usually used to describe some characteristic of a device, such as a transistor length, width, or Source/Drain areas and perimeters. Properties can be put on an instance to override the value defined in the symbol, or new properties can be created on an instance.

Annotation Graphics

Annotation graphics are non-electrical objects such as boxes, polygons, paths, and labels used to add comment or illustrations to the schematic.

Creating a Schematic View

To create a new schematic view, use **Cell > New View** (shortcut **N**). Simply select from the drop-down menus in each field. You can also use **Cell > New View** to create an entirely new cell.



Creating a New View (Cell > New View or “N”)

Design	Select a design file or library file from the drop-down list of those open where the cell will be saved.
Cell	Enter a cell name.
View type	Select the view type, i.e. schematic.
View name	Specify a view name.
Interface name	Specify which interface the new view belongs with.
Disable name validation	Check this box to prevent S-Edit from checking for naming violations.

Creating Instances

Instances in a cell are references to other cells. These cells might be common library cells such as basic circuit elements or larger, custom-designed cells. Instances are dynamically linked to their source cell so that any change you make to a source cell is reflected in each higher-level instance of that cell.

Updating Properties as You Place an Instance

Any option you set in the originating cell will be the default for all of its instances. However, you can override this setting in any of the instances, and the change will affect that instance only.

If you have a universal symbol library, you can use a TCL variable to automatically set process-specific properties as you place instances with **Cell > Instance** (see “Automatically Setting Properties During Cell > Instance” on page 156.)

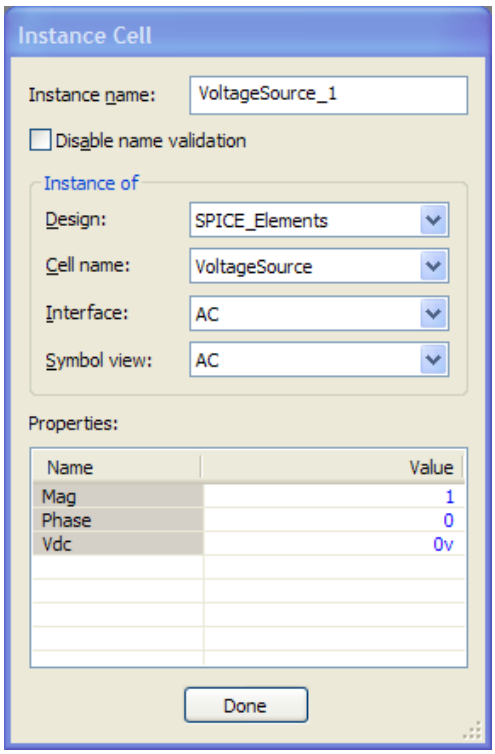
How To Instance a Cell (Cell > Instance or “I”)

[1] Highlight the cell in the **Libraries** cell list.

[2] Select a cell to instance by:

- using the **Instance** button in the Libraries navigator,
- clicking and dragging a cell from the Libraries navigator into the design area,
- using the **instance** icon on the toolbar , or
- using the menu command **Cell > Instance (**shortcut **“I”)**

[3] Property values inherited from the symbol can be overridden on a per instance basis by changing their values in the **Instance Cell** dialog before placing the instance. Modified values in the instance cell dialog will persist for all subsequent instances unless modified again.



[4] Drag the instance with the mouse and click in the design area to place it.

You will continue to instance the same cell until you either click **Done** in the **Instance Cell** dialog, double-click on the design area, choose a different cell to instance, select a new tool, or choose the **select** arrow.

Instances can be flipped horizontally or vertically prior to placement by pressing the “**H**” or “**V**” key, and can be rotated prior to placement by pressing the “**R**” key. Properties for each instance can be modified in the **Instance Cell** dialog prior to each placement.

Updating the Name as You Place an Instance

When you instance a cell, the instance name will be automatically assigned as the cell name with a whole number suffix incremented by one; for example, NMOS NMOS_1, NMOS_2, etc.

If you prefer, you can predefine an instance name prefix, for example MN, so when you instance NMOS, the instance names will instead be MN1, MN2, etc. Use the “InstanceName” property on the symbol to enter the instance name prefix. If there is no value in this field, S-Edit will default to using the cell name.

You can change the cell being instanced by selecting a new **Cell name** in the **Instance Cell** drop-down.

Instance Name
Enter a name for the instance. If no name is entered, S-Edit will by default use the *cellname_x*, where x is an integer that increments automatically.

However, if a value is entered in the **InstanceName** property field on the cell symbol, S-Edit will use that value as the instance name instead, as *instancename_x*, where x is an integer that increments automatically.

Disable name validation
Check this box to prevent S-Edit from checking for naming violations.

Instance of:
- **Design**— Select a library from those loaded to the active design.
- **Cell name**— Select a cell from those loaded to the design.
- **Interface**— Select an interface from those defined for the cell.
- **Symbol**— View the list of those open.

Properties
You can change the properties of the cell you are about to instance. This will be local override over the symbol default values.

Done
Click **Done** to stop instancing.

Locking and Hiding Instances

You can prevent edits to all instances in a design by disallowing instance selection. Similarly, you can set S-Edit so that all instances are hidden.

To set selectability and visibility, right-click on the instance toolbar button. As with other drawn objects, diagonal hatching on the icon indicates that instances are not visible, and a red dot in the lower left corner of the icon means that instances cannot be selected.

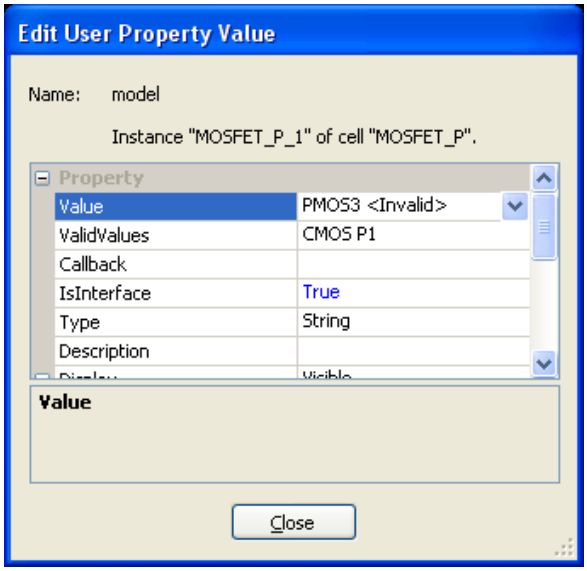
Note that even when S-Edit is set so that instances cannot be selected, when an instance is placed (using **Cell > Instance** or by dragging from the design navigator) it will remain in the selected state so it can be moved to the desired location. Then, once you place and unselect the new instance it will no longer be selectable.

Editing Instance Properties

To add a new type of property, see “Adding User Properties” on page 153.

Editing Properties from the Work Area

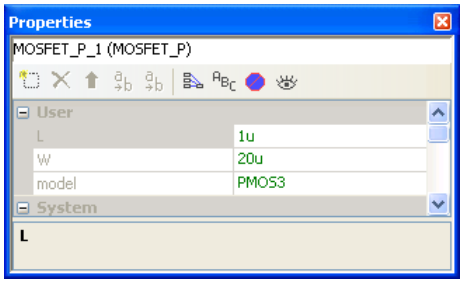
You can edit a property directly from the work area by using **Ctrl + click** to select it and then **Ctrl + double-click** to open the **Edit User Property Value** dialog. Only one object may be edited at a time from this dialog.



When you select an instance property from the design area, you will see a selection box when it is active in the Properties navigator. Changes made in the Properties navigator will update all instances if multiple instances are selected (see “Editing User Properties—the Properties Navigator” on page 151).

Editing Evaluated Properties

When the **Display Evaluated Properties Button** is on, properties display their evaluated value in the Property Navigator. Evaluated properties cannot be edited in the Properties navigator, if their **Editable** sub property evaluates to False.



Moving Instance Properties

Properties can also be moved in the design area. Use **Ctrl + click** to select a single property then use the middle mouse button to move it. Or, **right-click** on one property (it will not appear to be selected) then use **Ctrl + right-drag** to select multiple properties. Once selected, you can use the middle button to move them.

Selecting All Instances of a Cell

To quickly select all instances of a particular cell, select that cell name in the Libraries list and press the **Find** button. S-Edit will find and select each instance in the active work area.

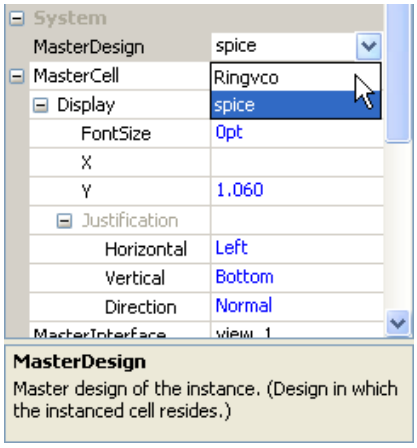
Editing Properties on Multiple Instances

You can change the properties on multiple instances at the same time. To do this, simply select all instances you want to change, then modify the desired values in the Properties navigator. You may modify any property value in this manner.

To select multiple properties, **right-click** on one property (it will not appear to be selected), then **Ctrl + right-drag** to select multiple properties (selection boxes will be visible at this point).

Changing Instances of a Cell to Instances of a Different Cell

To change all instances of one cell to instances of another cell, select the cells you want to change, then select a new cell from the **Master Cell** property under System properties in the Properties navigator. If the new cell you want to use is in a different library, you should select the library first from the MasterDesign property.



Setting the Visibility of Properties on Instances

Using the Display sub-property, you can also chose to show a property's name and value, its value only, or to hide the property altogether.

Note that property values are only hidden on instances in schematic view; they will always be visible in a symbol view.

Making and Labeling Connections

Connectivity is defined in terms of nets. In the most general sense, a net is created when one or more ports, labels, or wires are connected.

Nets derive their names from the ports or net labels to which they are attached. Connections formed by net labels exist only within a particular cell. Connections formed by ports can extend outside the cell.

Drawing Wires

To draw a wire, click on the **Wire** ( ) mode button on the Electrical toolbar then select one of the wire types ( ).

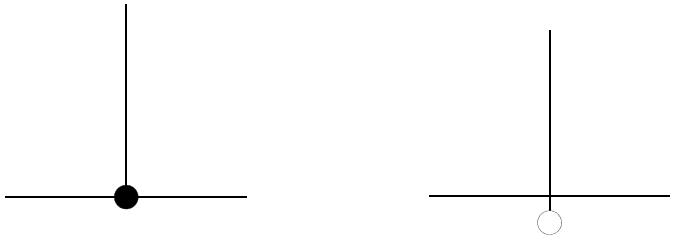
Click the DRAW (left) mouse button to place the beginning point of the wire. Continue to click with the left mouse button where you want to place a vertex. To end the wire segment, click the END (right) mouse button, or double-click the DRAW (left) mouse button. Either control will place the endpoint and finish drawing the wire.

Connections Points

A *connection point* is defined as an end of a wire or a bus, an instance of a port, a global or local port, a net label or a net cap.

Connection points that coincide are considered connected and are shown as filled-in circles unless they occur at the edge of an instance. Connections are usually explicit—for example, a wire end connected to a port, or two ports or two wires connecting directly to each other. Note that it is possible to form an implicit connection. For example, if you instance a cell that has two or more symbol ports named **A**, anything you connect in the instance to any port corresponding to **A** will also be implicitly connected to other pins corresponding to **A**.

An open connection point is one that does not coincide with another connection point. Unconnected ports and unconnected wire ends are shown as unfilled circles.

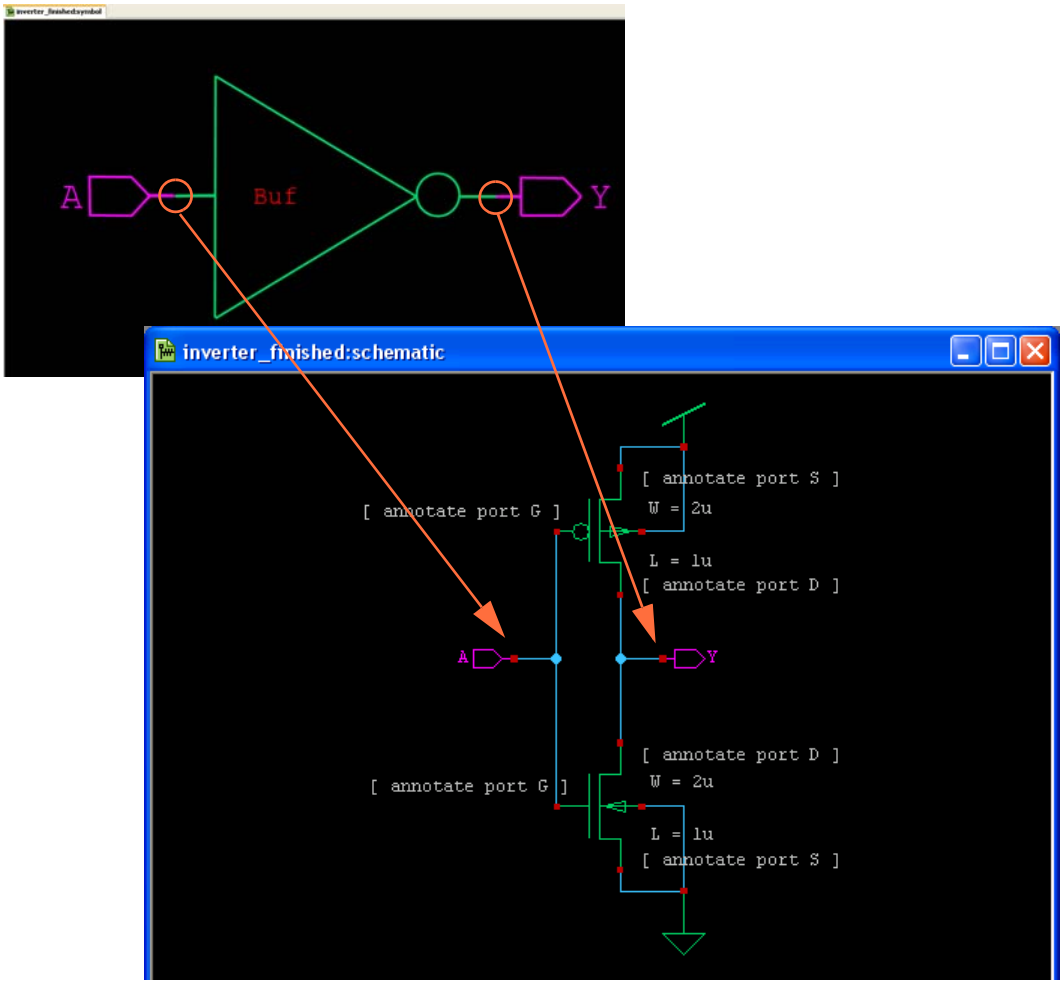


Connected Unconnected

If you click repeatedly on a connection point you will cycle selection through the wire segments that attach to it.

Hot Spots

A *hot spot*, drawn as a small red square, indicates a port connection in schematic view. Hot spots also assist with routing, as a mouse click within a hot spot will automatically align and connect wires.



Warning: Increasing the hot spot size may change your schematic's connectivity because each wire that intersects a hot spot will be connected to that spot. Any wire completely inside a hot spot will be removed. It is best not to set it a size larger than the original snap grid size. We recommend you back-up your design using the **File > Save > Save a Copy of Selected Design/Libraries** before proceeding.

Creating a Connection where Wires Intersect

Intersecting wires in S-Edit are *not* connected unless you explicitly add a connection point, using the solder point tool (shown below). Note that solder points must coincide with snap grid points.

Rubberbanding and Disconnecting Wires

Rubberbanding is the characteristic in S-Edit that preserves connectivity in a schematic view when an object is moved or edited. It is the default behavior.

If an object or wire with an open connection point is created, moved or edited, S-Edit will occasionally split a wire to create a new connection. This can occur if pre-existing connection points fall on each other or when a selected wire is split due to an edit.

You can detach rubberbanding before moving an object by using **Draw > Force Move** (shortcut key **Alt + M**).

To detach a wire from a pin, or to reattach wires that have detached because of an edit (for example, one moving a pin off-grid) use **Draw > Electrical > Connect/Disconnect**.

Connect/Disconnect is a toggle command that instantly attaches or detaches wires that fall within the perimeter of the tool. If you click on a connected wire the wire will disconnect, and vice-versa. It is also a non-sticky command—that is, it goes back to the previously selected tool after its execution.

When you click using the **Connect/Disconnect** tool S-Edit searches within a square area with sides equal to two times the snap grid size.

If there are any open solder dots on or inside the square they will be connected. If not, any connected wires will be disconnected by clipping them at the edge of the square. (With the exception of open connections from port instances, if they are exactly on the corner of the square.) Note that there is a small chance that new unwanted connections will be made this way.

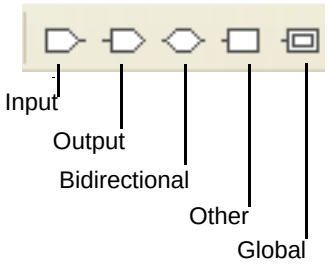
Connections are formed by drawing additional orthogonal wires connecting each open solder dot to the center of the square. If an angle is necessary it is created on-grid. When a connecting wire for a port instance requires an angle, the first segment will run perpendicular to the MBB of the instance rather than parallel. When an orthogonal wire patch is needed, the first patch segment will be colinear to the existing wire.

Adding Ports

Ports define the connection points that can be made to a symbol when the symbol is instanced. When instanced, ports do not display their name, but appear as open *connectivity circles* for attachment to wires. When connected to two or more items, they appear as filled circles; when connected to one item, they disappear. Ports exist in both symbols and schematics, and you can create and modify ports in both views.

Types of Ports

Five types of ports are available: Input, Output, Bidirectional, Other and Global, as shown in the toolbar below.



S-Edit treats the first four ports similarly. Port types imply different connectivity rules, enforced by the design checker. These rules are:

- All ports with the same name must have the same type on schematic and symbol
- A net on a schematic may not connect output ports of two instances
- A net on a schematic may not consist exclusively of input ports of instances

Global ports, however, have a different function—they are used to make global nets, which create connectivity throughout a design.

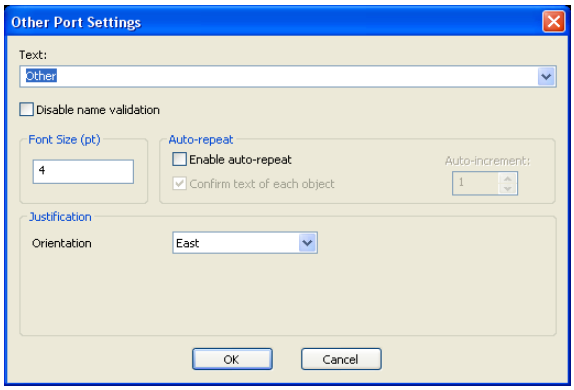
Drawing and Labeling Ports

To draw a port, select one of the port tools, and click in the design area. S-Edit will open the corresponding **Port Settings** dialog.

Note that you can use the **R**, **H** and **V** shortcut keys (for rotate 90°, flip horizontally and flip vertically) to change the orientation of a port before clicking to place it in the work area. This, rather than using the Properties navigator, is the suggested method.

Use the right mouse button or press the **ESC** key to exit port placement mode.

Draw > Electrical > {port type} Port



Text Enter the port label text in this field.

Font Size Enter the text size.

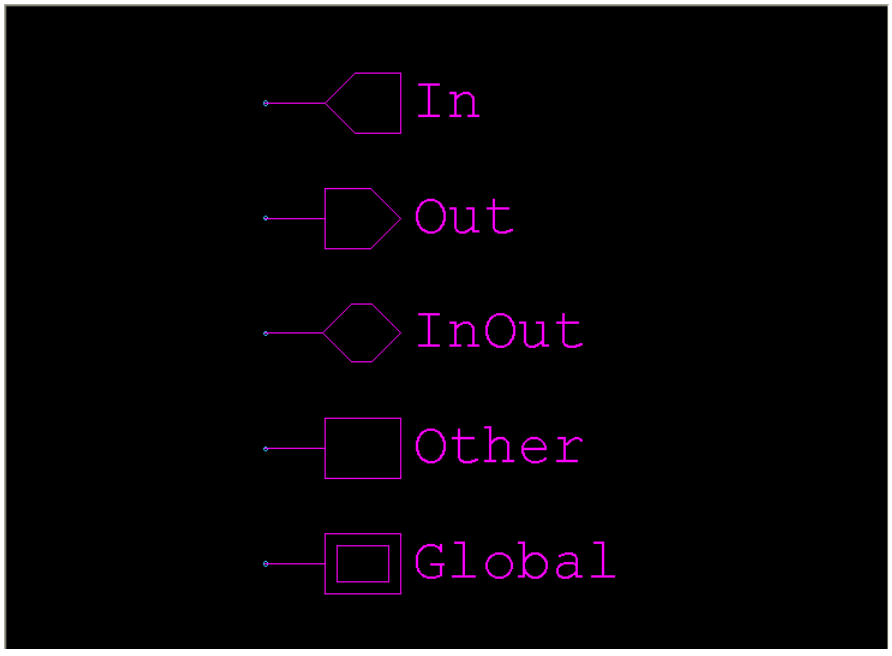
Auto-repeat If **Enable auto-repeat** is not checked, S-Edit opens the **Port Settings** dialog just once, places the port when you click **OK**, and reverts to **Select** mode.

With **Enable auto-repeat** checked, you can place ports repeatedly by clicking in the work area, without having to re-open the port settings dialog each time. S-Edit remains in port mode until you select a different mode or press **ESC**. The following controls may be added:

- **Confirm text of each object**—when checked, re-opens the **Port Settings** dialog each time a port is placed so you can enter changes to the next port placed.
- **Auto-increment**—when **auto-repeat** is active and the last character in the **Text** field is a numeral, S-Edit automatically increments the port label by the value entered here.

Justification Select **North**, **South**, **East** or **West,** where the cardinal directions describe the orientation of the port text with respect to the center of the connection point.

Each port type has its own graphic so it can be easily identified in the work area.



Naming Ports

In most cases, the port names on schematic views of a cell must match the port names on symbol views of that cell. Specifically, a cell is correctly defined when the arrangement of ports is such that:

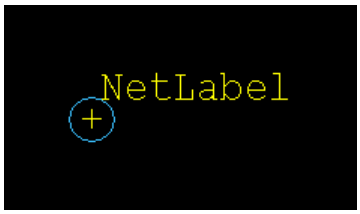
- For every non-global port on the symbol view, there exists a corresponding port of the same name and type on at least one of the cell's schematic views.

- For every non-global port in a schematic view, there exists a corresponding port of the same name and type on the cell symbol.

Attaching ports or net labels with the same name to differently named objects forms an implicit connection, even if those objects are not directly connected by wire. If you rename a port in one net, you break that implicit connection. Similarly, when a cell contains more than one net with the same name, those nets are connected, even if they are on cells.

Labeling Nets

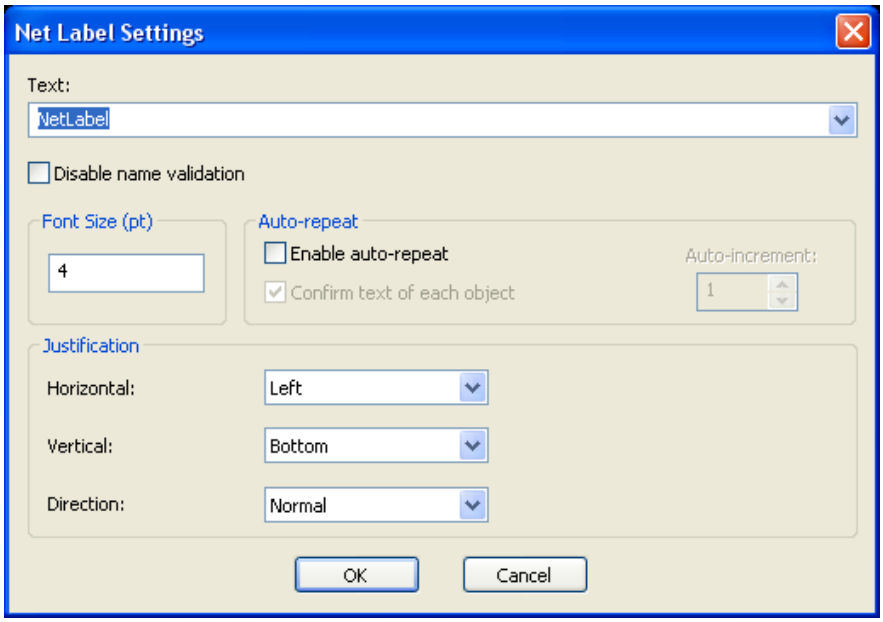
You can label nets to make your design more readable and to indicate connections within a cell. Connections formed by net labels exist only within a particular cell. When different cells contain nets with the same name, those nets are generally unconnected, with the important exception of global nets (see "Global Nets" on page 131).



Example of a net label with West orientation.

Click on the **Net Label** tool ( ) or use **Draw > Electrical > Net Label**. S-Edit will open the **Net Label Settings** dialog.

Note that you can use the **R**, **H** and **V** shortcut keys (for rotate 90°, flip horizontally and flip vertically) to change the orientation of a net label before clicking to place it in the work area.



Text Enter your net label text in this field.

Font Size Enter the text size.

Auto-repeat

If **Enable auto-repeat** is not checked, S-Edit opens the **Net Label Settings** dialog just once, places the net label when you click **OK**, and reverts to **Select** mode.

With **Enable auto-repeat** checked you can place net labels repeatedly by clicking in the work area, without having to re-open the net label dialog each time. S-Edit remains in net label mode until you select a different mode or press **ESC**. The following controls may be added:

- **Confirm text of each object**—when checked, re-opens the **Net Label Settings** dialog each time a net label is placed so you can enter changes to the next label placed.
- **Auto-increment**—when **auto-repeat** is active and the last character in the **Text** field is a numeral, S-Edit automatically increments the net label text by the value entered here.

Unnamed nets are given the default prefix N_*x* where *x* = 1, 2, 3, etc. However, you can use the Tcl variable “tanner_unnamednetprefix” to assign your own prefix.

Justification

All text orientations are with respect to the origin point of the label, indicated by a cross.

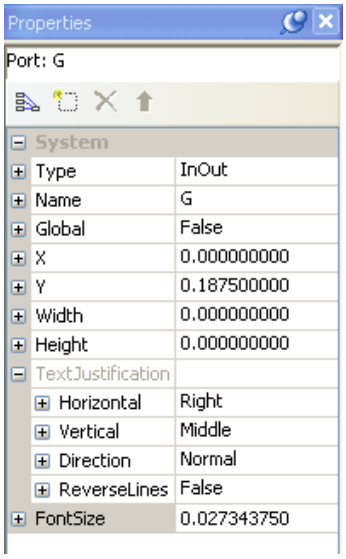
Horizontal: select **Left**, **Center** or **Right** horizontal text alignment.

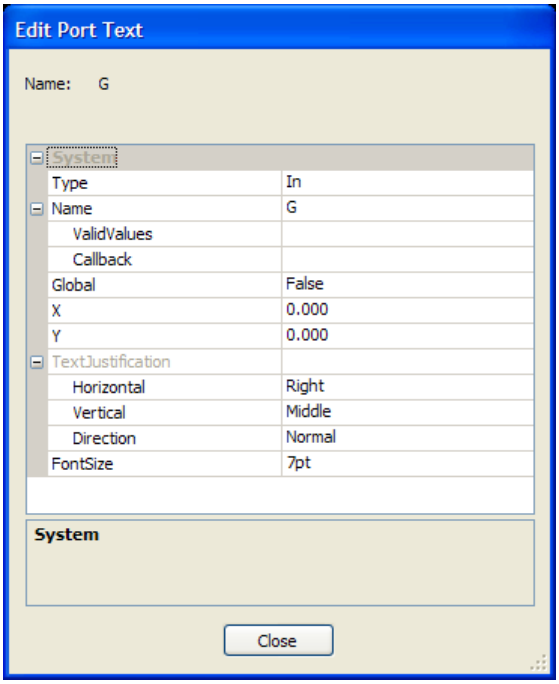
Vertical: select **Top**, **Middle** or **Bottom** vertical text alignment.

Direction: select **Normal**, **Upside down**, **Down** or **Up** for the text direction.

Editing Port Properties

Ports, net labels, and comment labels may be edited in the Properties browser. Sub-property values such as text size or justification may also be changed. Click on the desired port to display or change its values in the Properties browser or double-click on a port to open the Edit Port Text dialog.





Buses, Bundles and Arrays

S-Edit supports buses, arrays and net bundles. A *net* is the fundamental single unit of connection. A *bus* is a set of connections with the same name, plus a numerical identifier and increment which is the syntax that defines it as such. Similarly, an *array* is a set of ports within an instance, with the same name plus a numerical identifier and increment that defines related multiple connection points.

A *bundle* is collection of both nets and buses, where the nets do not need to share a name with the bus. A *port bundle* is a port defined to accept more than one wire, defined as such by the same numerical identifier and incrementing syntax that defines a bus.

Buses in S-Edit can be grouped or nested to any required depth. If multiple buses with different dimensions connect to a single wire, you have the option to treat this as multiple devices in parallel or to treat it as an error.

Creating a Bus

You create a bus with the net label tool by using a special naming syntax that specifies how many nets are in the bus, and how their number increments. The default increment value is one and may be omitted if that is the desired value.

For example, UpData<0:7:2> creates a bus four bits wide, with nets UpData<0>, UpData<2>, UpData<4>, ... UpData<6>.

You can create two-dimensional buses by naming a wire *busname* <n1:n2:stepa><n3:n4:stepb>. As with arrays (see below), the second range increments before the first range.

Repeating Names and Grouping Buses

S-Edit supports the prefix notation <*n>, grouping notation ( ) and suffix notation <*n> for buses as follows.

Prefix Notation

You can repeat a single signal name, a group of signal names, or a vector term any number of times in the wire name by placing a prefix repeat operator <*n> in front of the name, where ***n*** is a positive integer that defines the number of times to repeat each bit in the vector term.

Use the prefix repeat operator <*n> to repeat a single-signal name. For example, the following equivalent wire names both describe the same four-bit wire:

```
<*2>A,B,C
A,A,B,C
```

Use the prefix repeat operator <*n> and parentheses to repeat a group of signal names. For example, the following two wire names are equivalent:

```
<*2>(A,B),C
A,B,A,B,C
```

You can also use combinations of the prefix repeat operator <*n> and parentheses to nest parenthetical expressions to any required depth. Nested expressions are expanded from the innermost expression outward. For example, a name with the expression

```
<*2>(A,<*2>(X,Y)),B
```

expands to

```
A,X,Y,X,Y,A,X,Y,X,Y,B.
```

Suffix Notation

You can also use <*n> as a suffix repeat operator after a vector term, where n is a positive integer that defines the number of times to repeat each bit in the vector term.

For example, use the suffix repeat operator <*n> to repeat each bit in a group of bit names before expanding the vector term. For example, the following three names all describe the same six-bit wire:

```
A<0:2*2>
A<0*2,1*2,2*2>
A<0,0,1,1,2,2>
```

Or, use the suffix repeat operator <*n> and parentheses to repeat the sequence of bit names. In this case, the vector term is expanded before the bits are repeated. Again, the following names all describe the same six-bit wire:

```
A<(0:2)*2>
A<0:2,0:2>
```

```
A<0,1,2,0,1,2>
```

Expanding Buses

S-Edit supports expansion of bus pins with the colon (**:**) notation, where ":" indicates a range of pins. For example, a cell "Dig2" has a symbol with properties:

```
SPICE.DEFINITION = .subckt $Cell (%{Data<3:0>}) Vdd Clk (%{Out<0:3>})
SPICE.OUTPUT = X${Name} (%{Data<3:0>}) %{Vdd} %{Clk} (%{Out<0:3>})
   $MasterCell %(Data<3:0>)
```

SPICE.DEFINITION will expand buses in the subcircuit definition line and SPICE.OUTPUT will expand buses in the netlist. When a cell containing an instance of Dig2 is exported, we will get:

```
.subckt Dig2 (Data<3:0>) Vdd Clk (Out<0:3>)
.ends

subckt Dig2 (Data<3> Data<2> Data<1> Data<0>) Vdd Clk (Out<0:3>)
```

Creating an Array

You create an array by applying array syntax when you name an instance. So, an instance named *array_name*<n1:n2:*step*> creates an array of instances named array_name<n>, where n starts at n1, ends at n2, and increments by the *step* value. The step increment is one by default and can be omitted.

Two dimensional arrays may be created by naming an instance U<n1:n2:step1><n3:n4:step2>, where the second range increments first.

For example, the instance name U<0:7><0:3> creates an array of instances named:

```
U<0><0>, U<0><1>, U<0><2>, U<0><3>,
U<1><0>, U<1><1>, U<1><2>,  U<1><3>,
U<2><0>, U<2><1>, U<2><2>,  U<2><3>,
...
U<7><0>, U<7><1>, U<7><2>,  U<7><3>
```

When you instantiate a device as an array and that device has annotation parameters such as [annotate port IN]), S-Edit will be unable to find the proper connectivity view. Related errors are displayed using question marks to encode the messages as follows:

???— the annotation failed due to lack of simulation results

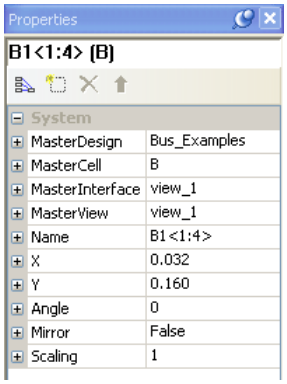
?— there is no such port or no such instance

(blank)— there is no connectivity view or extract has not been performed

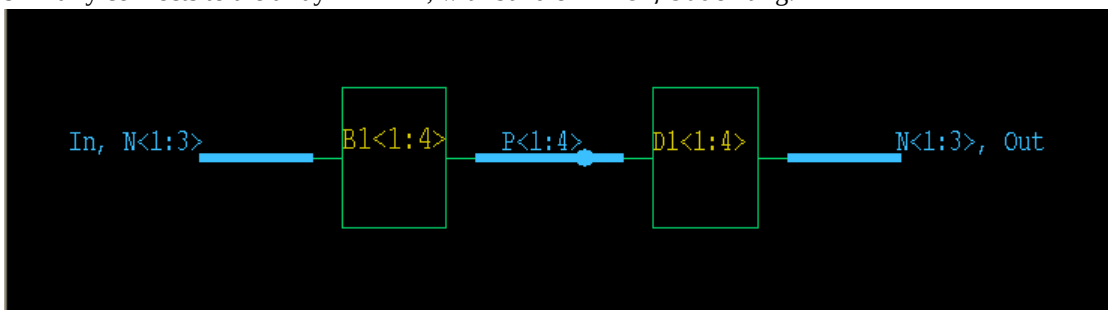
Alternately, you can override the visibility of those properties on array instances.

Array Example 1

In the following example, we have created an array of four cells by giving an instance of cell B the name **B1<1:4>**.

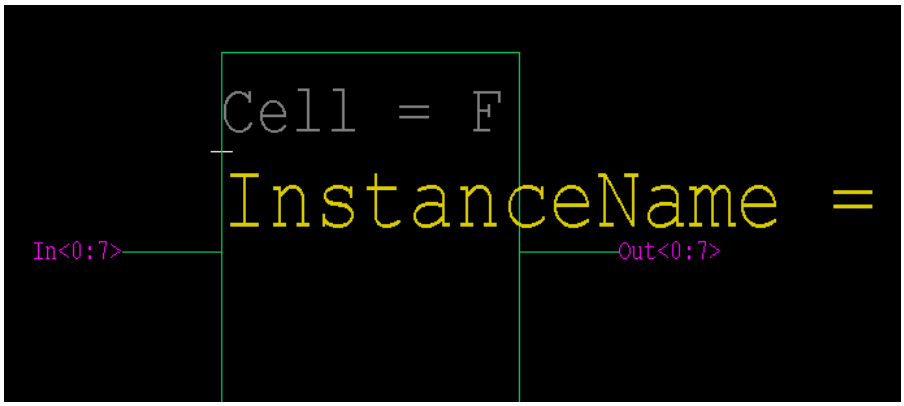


The bundle **In, N<1:3>** connects four wires **In, N<1>, N<2>, and N<3>** to array **B1,** and bus **P<1:4>** similarly connects to the array **D1<1:4>**, with bundle **N<1:3>, Out** exiting.



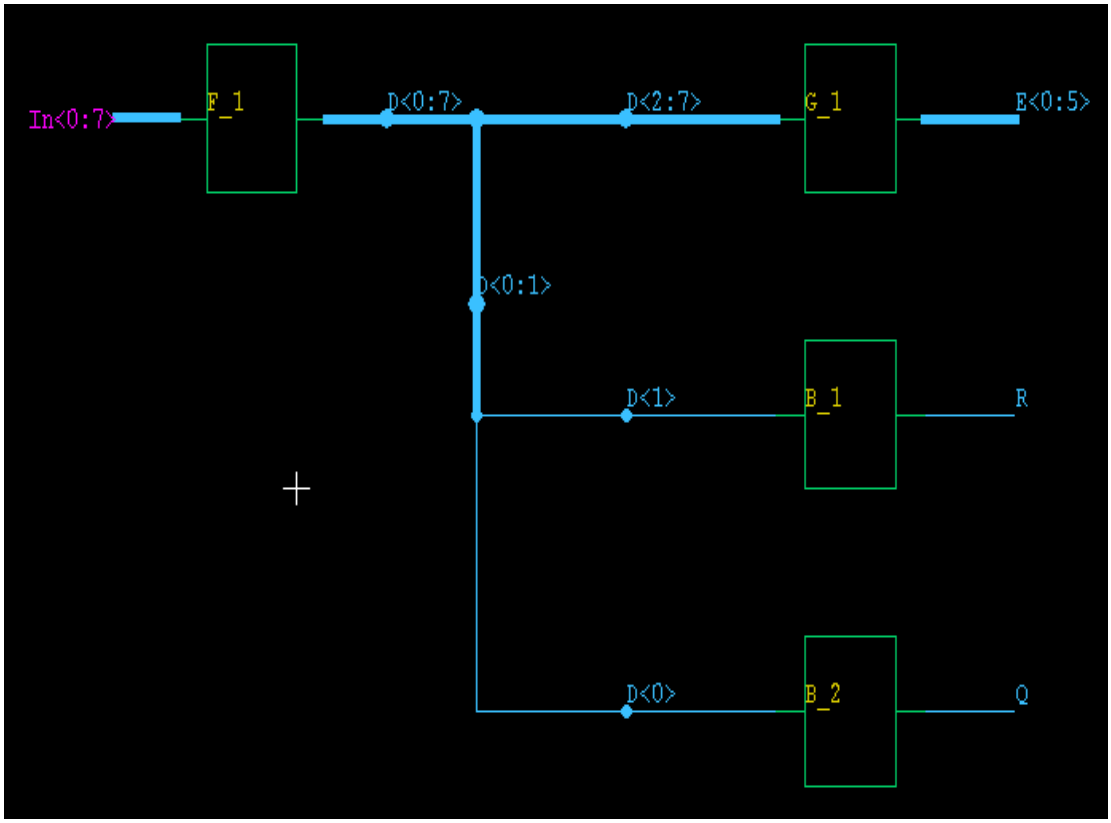
Array Example 2

In this example, cell **F** contains port bundles **IN<0:7>** and **Out <0:7>** so that it can connect to eight wires.



Buses **In<0:7>** and **D<0:7>** form an eight bit wide bus, respectively, in and out of instance **F_1**, which splits so that wires **D<2>**, **D<3>**, **D<4>**, **D<5>**, **D<6>**, and **D<7>** go into instance **G_1** with five-bit port

bundles in and out to accommodate **D<2:7>** and **E<0:5>.** The single wire **D<1>** goes into instance **B_1** and out as net **R**, and single wire **D<0>** goes into instance **B_2** and out as net **Q**.



Global Nets

Global nets simplify the drawing and maintenance of electrical schematics. When a net is global, you can connect or disconnect it throughout a design without drawing or deleting wires. Global nets are especially useful for power, ground, clock, reset, and other circuitwide nets that require routing throughout a design.

Like standard nets, global nets connect across all of the schematic views of a cell. Unlike standard nets (which connect to nets outside the cell only through ports), global nets automatically connect between all cells in a design (subject to certain scoping rules; see “Naming Global Nets” on page 133). Any changes made to a global net’s name or state propagate throughout the design.

There are two ways to create a global net. You can add a global port to a schematic page, or you can instance a “global symbol”—that is, a symbol which has a global port or has one on it’s corresponding schematic page.

Global Ports

Global ports indicate connection points for global nets. Unlike regular ports, the names of global ports are significant outside of the cell where they are placed. The name of a global port is the default name of the global net associated with it. For example, if you want to create a global net with the default name Gnd, name the global port Gnd.

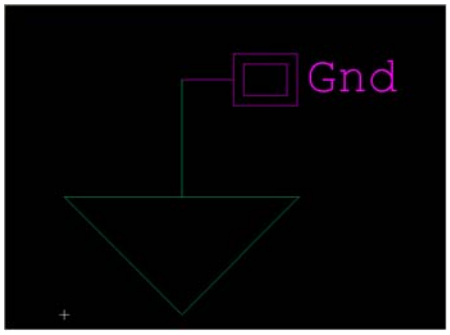
To place a global port, use the **Global Port** tool (See “Adding Ports” on page 122.)

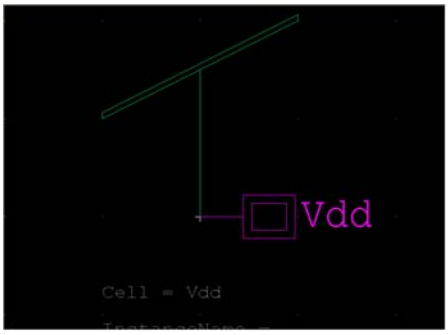
Global Symbols

Global symbols are special cells that function as wireless connectors. They are defined as such when they contain one or more global ports.

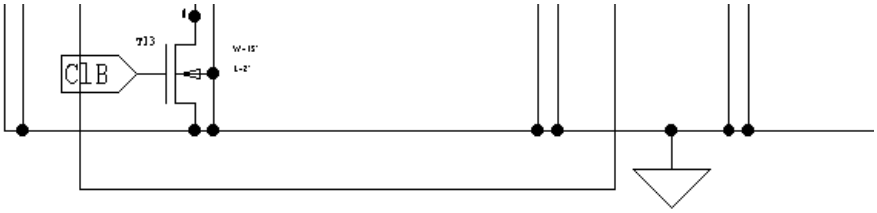
When you attach a global symbol to a net, you connect that net to all other nets on every view and cell in the design file that are attached to a like global symbol. Such nets then become global nets. Conversely, a net ceases to be global when all global symbols are detached from it. You can add any number of new global symbols to a design.

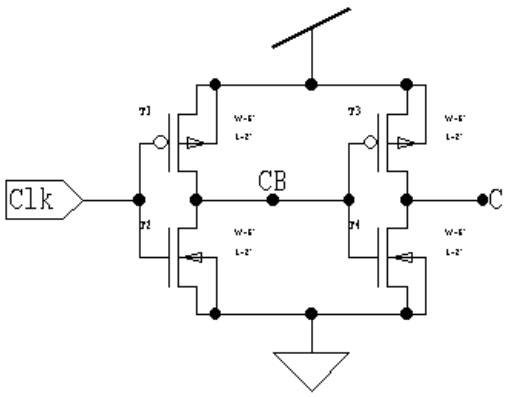
Sample files shipped with S-Edit contain symbols for **Gnd** and **Vdd** as shown below.





In the figure below, two Gnd symbols are attached to two different wires in a cell. Since Gnd is a global symbol, the two wires are thus connected to each other. In addition, any wires in any other cell which has the Gnd symbol attached is also connected to these nets.





Naming Global Nets

The default name of a global net is the name of the global port in the symbol that defines the global net. An uncapped global net can never be renamed, and always uses the default net name; if you place a port or net label on an uncapped global net, it will automatically acquire the name of the global net.

You can rename a capped global net using a port or net label. The new name affects the current cell only. When you export a netlist, however, the new name will affect any cells containing uncapped global nets which are instanced in the current cell. By renaming capped global nets, you can use the same global symbol to represent different nets with different names.

propagated	A global port or an instance containing one appears on the cell, with no net caps on the associated global net. The net propagates up and down the cell hierarchy.
capped	A global port or an instance containing one appears on the cell, with a net cap named so it is associated with the global net. The net does not propagate up the cell hierarchy.
hidden	A global port or an instance containing one does not appear on this cell, but global net propagation “passes through” this level of the hierarchy because the global port appears, uncapped, in one of the cells instanced within the current cell. In SPICE output, these nets will actually appear in the cell as “placeholder” nets. You can connect to a hidden net by placing a net label (or port) in the cell with the same name as the hidden net (or by instancing the global symbol).

When global symbols collide, S-Edit will arbitrarily choose one of their names as the new name for the net. There is no guarantee that this net name will be stable; it may change when further edits are performed.

If a global symbol collision causes two separate global nets to be connected, S-Edit will use the common new name to designate the net and all nets connected to all instances of all colliding symbols. Until the collision is resolved, all of these nets continue to be electrically connected. Avoid such situations wherever possible.

One way to avoid collisions between global symbols on the same net is to use a voltage source cell to set the voltage between the nets. Let a property of the cell be the voltage between the nets. Then edit the property to set any desired voltage between the nets, even 0 volts. You can find voltage source cells in the S-Edit sample libraries.

Effective Design with Global Nets

Here are some tips for the effective use of global nets in your design.

- Avoid using several similar global symbols, such as **Vdd1**, **Vdd2**, and so on, for the purpose of separating similar global nets. Instead, use a single Vdd global symbol everywhere. Then place net caps at appropriate places to separate the different nets from each other. Finally, simply use net labels at the capped level to rename the nets. By using only one global port, you can instantly change the scope of these global nets without having to search for and replace different ports.
- Avoid placing global ports or net caps on a net that is connected via a port to higher levels of the design. The presence of a global port on such a net will automatically make its higher-level

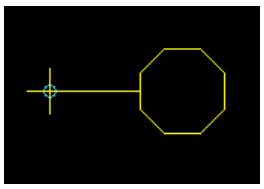
connections global, regardless of the designer's intention. Moreover, net caps will not prevent such a net from propagating. Be careful when naming different grounds.

- T-Spice considers all nets named Gnd, GND, gnd, and 0 to be connected. If any .subckt pin name is one of the names for ground (0, gnd, gnd!, or ground) then all ground references in the subcircuit will be aliased to that pin.
- When importing Cadence schematics, S-Edit imports interface ports and net labels that end with '**!**' as global ports.

Capping Global Nets

Global nets automatically propagate up the design hierarchy. For example, if the cell in the previous example is instanced in another cell, Gnd will exist in that upper cell.

Net caps halt, or “cap,” the propagation of a global net from subcircuit definitions that are lower in the hierarchy.



Graphic for a net cap.

You place a net cap with the **Net Cap** tool.

When a net cap is placed in a cell, the global net will not propagate to any cell that instances that cell. The net’s scope is limited to the current cell and any cells instanced on its schematic views. Any global nets with the same global symbol that exist outside the scope of the capped net may have the same name as the capped net, but are unconnected.

One important exception to this rule occurs when you export SPICE files from S-Edit for use in T-Spice and some other SPICE simulators, which treat all ground nets—in T-Spice, nets with any of the names Gnd, GND, gnd, or 0—as connected to 0.0 volts, regardless of S-Edit’s scoping rules.

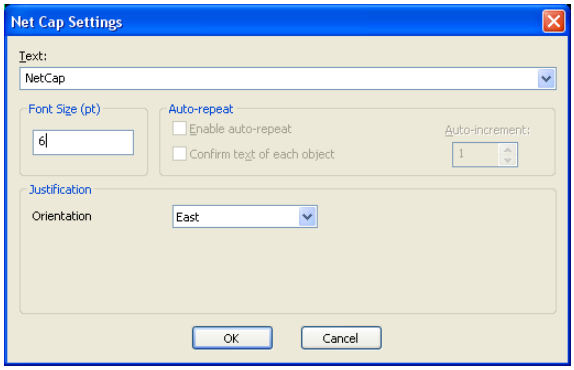
Note that the net cap does not have to be attached to a net—it can be placed anywhere in the work area. However, the name is critical as it defines the net cap.

Naming Net Caps

A net cap must be given the name of the net you wish to cap.

Click on the **Net Cap** toolbar button or use **Draw > Electrical > Net > Cap** and place the net cap anywhere on the schematic page where you want the cap to occur.

If a port is connected to a global net, the net propagates to the higher levels through the cell port connection, just as a regular net does. This happens even if the global net is capped. This is the only way a capped net can propagate to a higher cell.



Text — Enter the name of the net cap in this field.

Font Size — Enter the text size in points.

(Enable auto-repeat is not functional for this operation.)

Justification — Select **North**, **South**, **East** or **West,** where the cardinal directions describe the orientation of the text with respect to the center of the connection point of the cap.

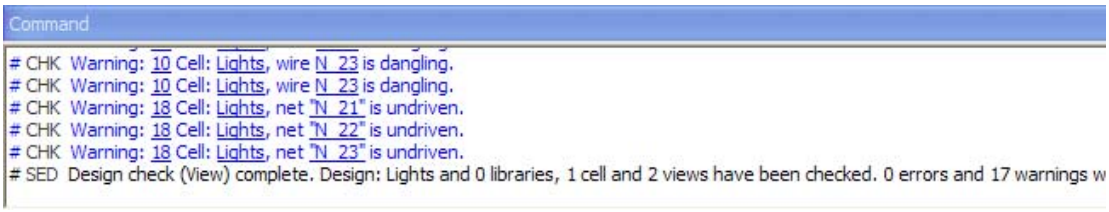
Checking a Design for Errors

The S-Edit design checker searches schematic pages for many of the mistakes commonly made during the schematic creation process.

S-Edit provides the option to check a schematic view, its hierarchy, a design or a full design and its libraries. Use **Tools > Design Checks > [***choose from the pulldown menu***]** (or the toolbar button) to initiate the check.

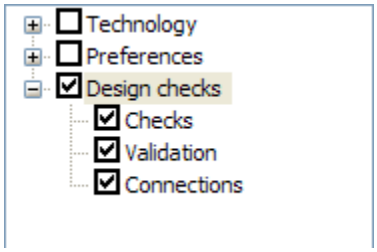
Design checks are performed automatically before EDIF, Verilog, VHDL, or SPICE export. If one or more errors occur, the export is aborted.

After a design check, warnings are displayed in the Command window. You can click on the error number to jump to that rule in the design checks setup dialog, where it will be highlighted.


Command
CHK Warning: 10 Cell: Lights, wire N_23 is dangling.
CHK Warning: 10 Cell: Lights, wire N_23 is dangling.
CHK Warning: 18 Cell: Lights, net "N_21" is undriven.
CHK Warning: 18 Cell: Lights, net "N_22" is undriven.
CHK Warning: 18 Cell: Lights, net "N_23" is undriven.
SED Design check (View) complete. Design: Lights and 0 libraries, 1 cell and 2 views have been checked. 0 errors and 17 warnings w

Design Check Setup

The **Save**, **Load** and **Close** buttons in design check setup work just as in the other setup dialogs (see "Saving and Loading Setup Options" on page 35). There are three design check setup pages, as shown below.



Setup > Design Checks > Checks lets you set the severity of individual layout checks to **Error**, **Warning**, or **Ignored**.

Setup > Design Checks > Validation sets checks for naming errors in layout objects such as views and instances.

Setup > Design Checks > Connections lets you individually configure the connections between pin and port types as **Error**, **Warning**, or **Ignored**.

Note: Note that existing cell names and names that are entered using the Properties navigator, are not validated until a file undergoes a design check. Similarly, no validation checking is performed when a TCL command is entered directly in the Command window.

Setup > Design Checks > Checks

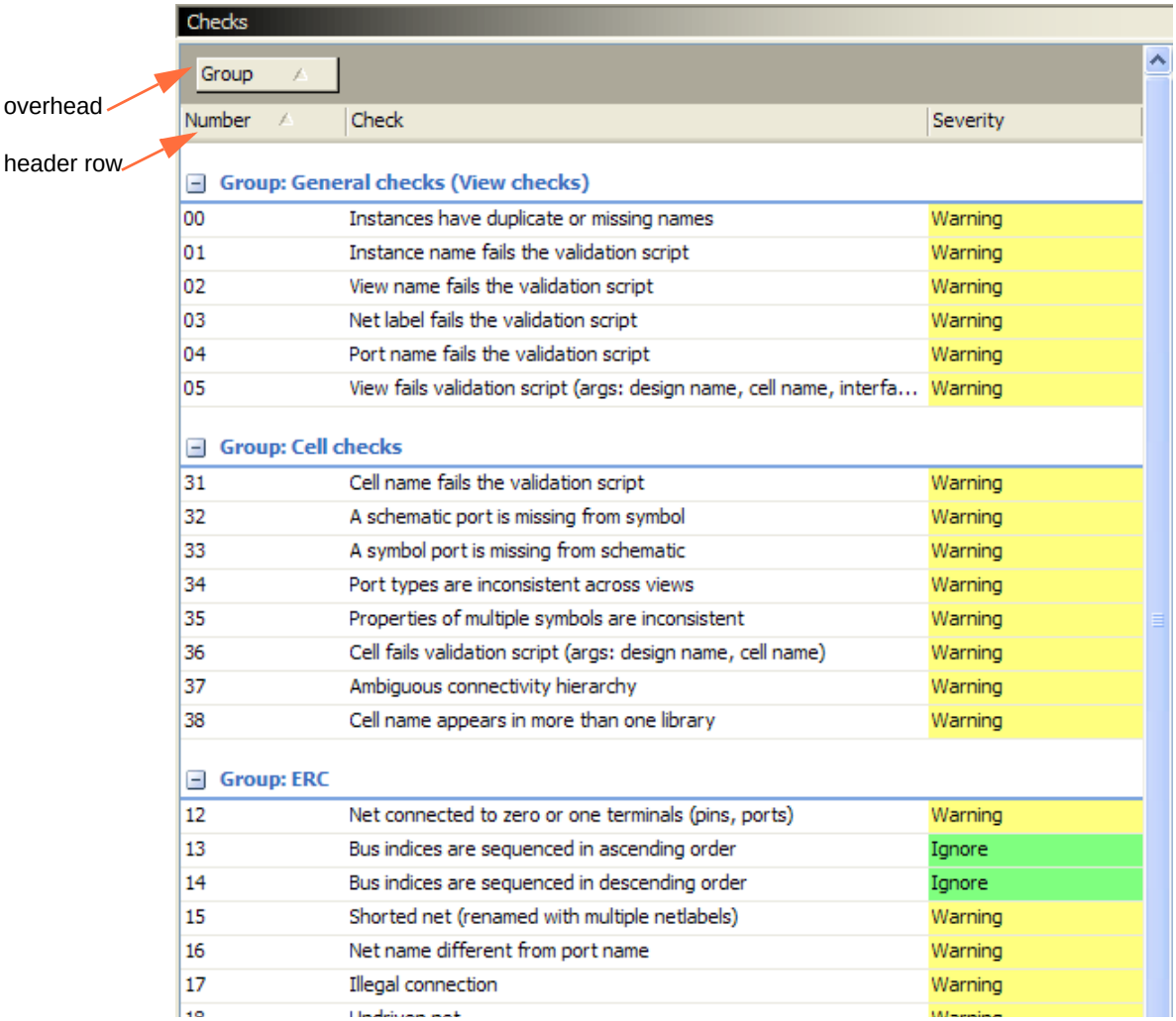
The checks that S-Edit performs when running Design Check are explicitly listed in **Setup > Design Checks > Checks**. Individual checks can be set for designation as an **Error**, which will prevent the proper connectivity from being formed, or a **Warning**, which will not prevent S-Edit from extracting connectivity but is likely to be unintended. An individual check also can be disabled by setting it to **Ignore**, in which case it is not checked.

You can set the maximum number of times that errors are printed for each individual design check, where that one value applies to all errors. If an error is displayed more than the set number of times, that error display is suppressed thereafter.

You set this limit with a Tcl variable set *tanner_maxdesigncheckmessages 9999*, where the value can range from Zero or a negative number, which means "show all errors" to a maximum count of 999 displayed. (Note that there can be serious performance consequences to increasing this limit, as the time required to log the warnings is often much greater than the time required to do the checks.)

- **Views**—*ex*. instances names are unique and pass the validation script.
- **Cell**—*ex*. correspondence of ports on symbols and schematic.
- Electrical rule checking (**ERC**)—*ex*. shorted or unconnected nets.
- **Placement** or **Physical**—*ex*. overlapping instances or dangling wires.
- **Verilog**, **VHDL** and **SPICE**—*ex*. illegal or colliding net names.

Use this dialog to set the error flagging for each specific design check.


overhead
header row
Checks
Group
Number
Check
Severity
Group: General checks (View checks)
00 Instances have duplicate or missing names Warning
01 Instance name fails the validation script Warning
02 View name fails the validation script Warning
03 Net label fails the validation script Warning
04 Port name fails the validation script Warning
05 View fails validation script (args: design name, cell name, interfa... Warning
Group: Cell checks
31 Cell name fails the validation script Warning
32 A schematic port is missing from symbol Warning
33 A symbol port is missing from schematic Warning
34 Port types are inconsistent across views Warning
35 Properties of multiple symbols are inconsistent Warning
36 Cell fails validation script (args: design name, cell name) Warning
37 Ambiguous connectivity hierarchy Warning
38 Cell name appears in more than one library Warning
Group: ERC
12 Net connected to zero or one terminals (pins, ports) Warning
13 Bus indices are sequenced in ascending order Ignore
14 Bus indices are sequenced in descending order Ignore
15 Shorted net (renamed with multiple netlabels) Warning
16 Net name different from port name Warning
17 Illegal connection Warning

Each of the columns is sortable. When a column header is in the overhead area, that category is the primary sort value and rules are secondarily sorted within each category.

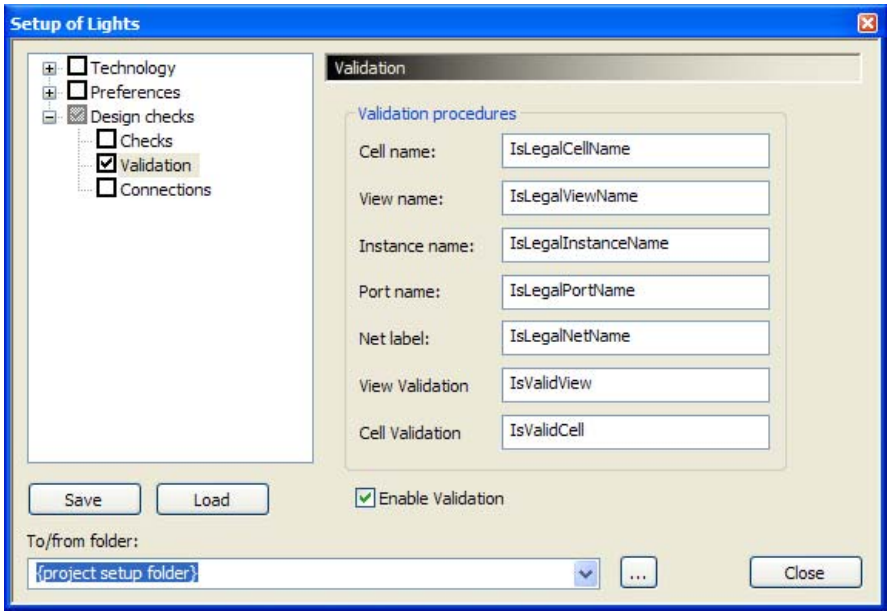
For example, in the default setting shown above, **Group** is the primary sort column and rules are sub-sorted by **Number** within each group. To sort rules regardless of group, drag all column headers to the header row.

Assorted Rule Notes

- Instances are considered to be overlapping if their bounding boxes overlap and the area of the overlap is greater than half the area of the smaller bounding box.
- S-Edit ignores symbols that do not contain any ports (explicit or global).
- **Setup > Design Checks > Checks > Group: Cell Checks > Cell name fails the validation script** controls a validation script in "**...Validation > Cell Validation**". The script in this field is a user script and is assumed to already be loaded by any of the usual way scripts are loaded, such as from open.design. The script is run on all cell views in the design when design checks are run.

Setup > Design Checks > Validation

S-Edit uses TCL functions to check for naming errors in the following objects: cells, views, instances, port, and nets and to perform user-defined checks on cells or views.



A default set of validation functions (**IsLegalCellname**, **IsLegalViewName**, **IsLegalInstanceName**, **IsLegalPortName**, **IsLegalNetName**, **IsValidView**, and **IsValidCell**) is provided with S-Edit in the **tannerinit.tcl** file. When you create a new design it is automatically initialized to point to these default functions.

While it is possible to overwrite the TCL functions in **tannerinit.tcl** Tanner EDA strongly recommends against it as the **tannerinit.tcl** file is overwritten with each product upgrade.

To create your own naming constraints you can write TCL functions, save them to the **C:/Documents and Settings/<username>/Application Data/Tanner EDA/scripts/startup** folder, and enter those function names in the fields of the **Setup > Design Checks > Validation** page. Contents of the S-Edit startup folder will take precedence over functions in **tannerinit.tcl**.

For example, you might want to prevent the use of spaces in cell names so that your design will adhere to GDSII naming conventions. You would write a TCL function and enter the name of that function in the **Cell Name** field of the **Validation** setup page.

When you enter a cell name in, for example, **Cell > New View**, S-Edit executes the TCL function referenced in the setup page, and displays any violations in the Command window, highlighted in red. For example, “Validation violation: <*description*>. Create anyway?” where *description* is the string returned by the validation function.

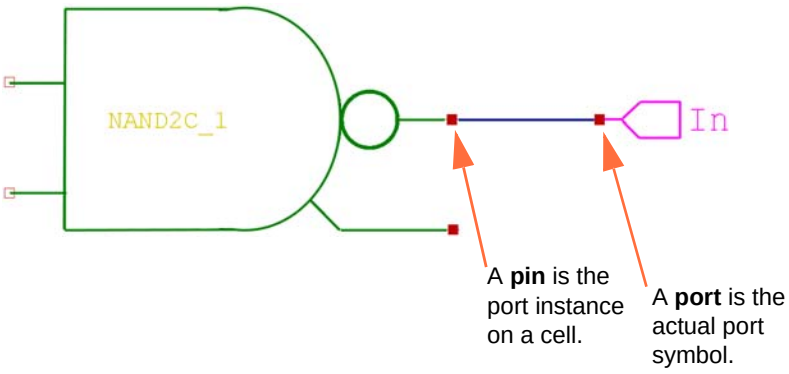
Note that validation is controlled in two ways—by object type, using the **Disable name validation** checkbox in the dialogs where you can enter a name, and globally from the **Validation** setup page using the **Enable Validation** checkbox.

Cell name Enter the name of the TCL procedure with the set of naming constraints you want to apply to cells.

View name	Enter the name of the TCL procedure with the set of naming constraints you want to apply to views.
Instance name	Enter the name of the TCL procedure with the set of naming constraints you want to apply to instances.
Port name	Enter the name of the TCL procedure with the set of naming constraints you want to apply to ports.
Net label	Enter the name of the TCL procedure with the set of naming constraints you want to apply to nets.
View Validation	Enter the name of the TCL procedure that will be applied to each instance.
Cell Validation	Enter the name of the TCL procedure that will be applied to each cell.
Enable Validation	Check this box to enable all TCL name validation functions; leave it empty to disable them all.

Setup > Design Checks > Connections

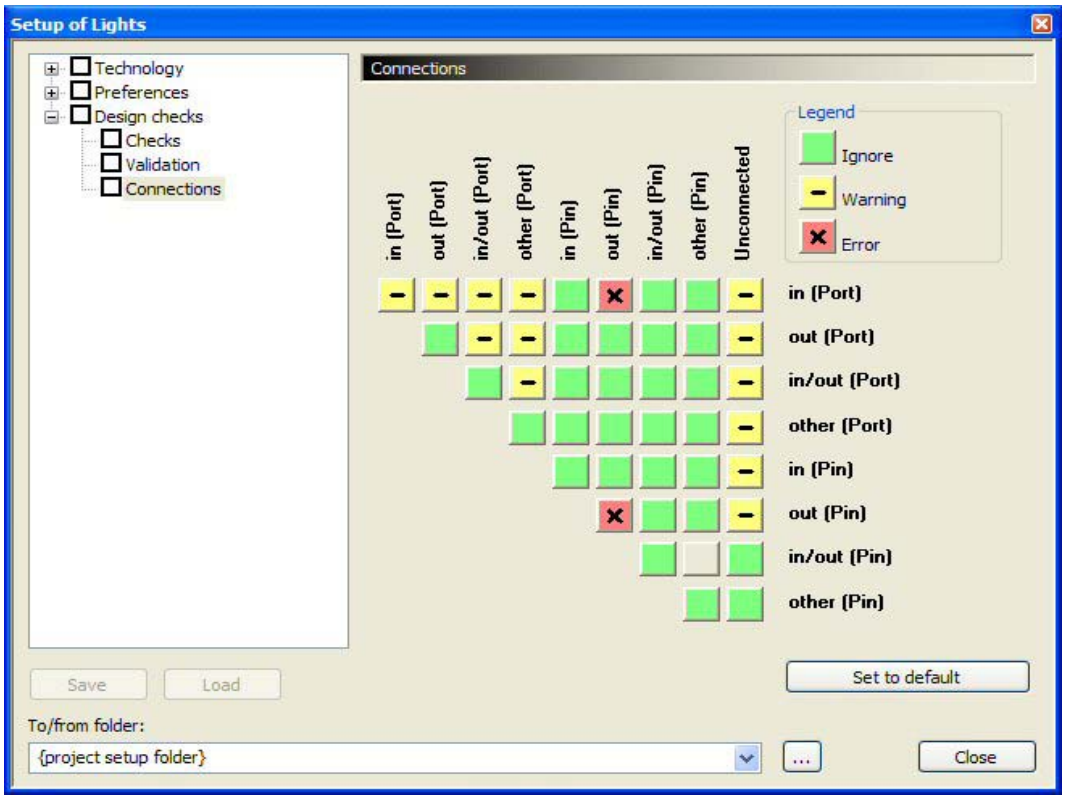
Use this matrix to set the level of warning for each combination of pin type, port type and unconnected state, where a **pin** is a port instance on a cell and a **port** is an actual port object.



Assorted Connectivity Recommendations

- We recommended that you do not give a net the same base name as a bus. For example, a net named **N** and a bus named **N<0:7>** could cause confusion.
- Nets ripped from a netbundle must actually exist in the netbundle.
- Bus dimensions must be compatible with rippers, portbundles, subscripts, etc. so you are not trying to rip a 5th wire from a 4-wire bus.
- Nets can have at most one OUTPUT portref connected to them.
- Nets must have at least one portref that is NOT of type INPUT connected to them, unless they are connected to an interface port of type INPUT

The icon indicates **Ignore**, the icon indicates **Warning** and the icon indicates an **Error**. Note that clicking on a button in the legend sets all connections to that state. **Set to Default** returns warnings to their initial state, shown below.



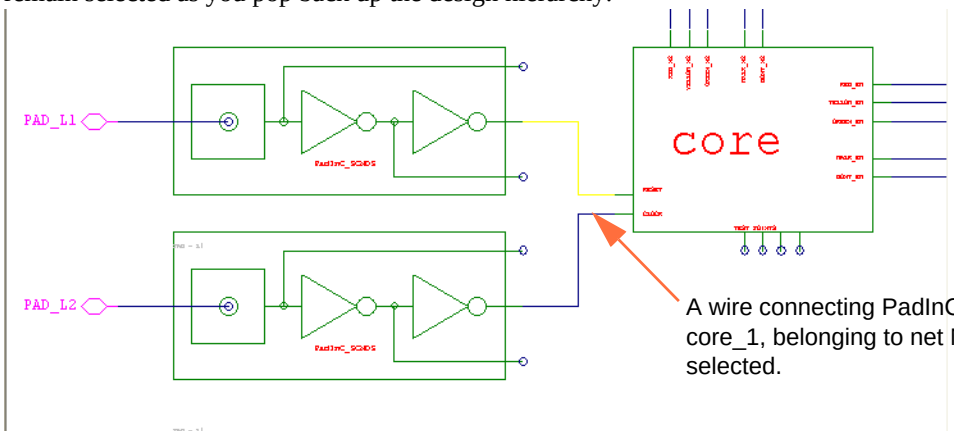
Net Highlighting

You can use the **Push into Context** and **Pop Context** buttons to trace a net through your design hierarchy.

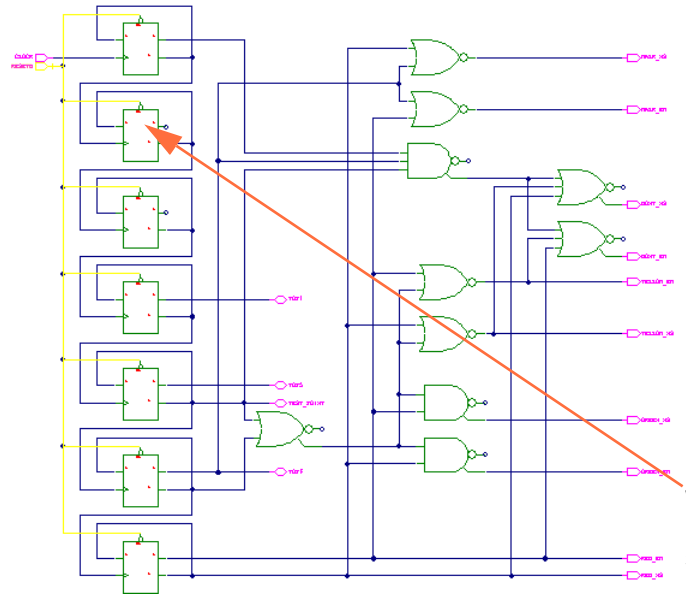
When you select a wire or port at the top level, the entire net it belongs to will be selected when you push into any instance to which it is connected. Unless you select part of a different net, the net will remain selected as you push down to the primitive level.

Net highlighting from a schematic netlist selects the port instance when there is no net.

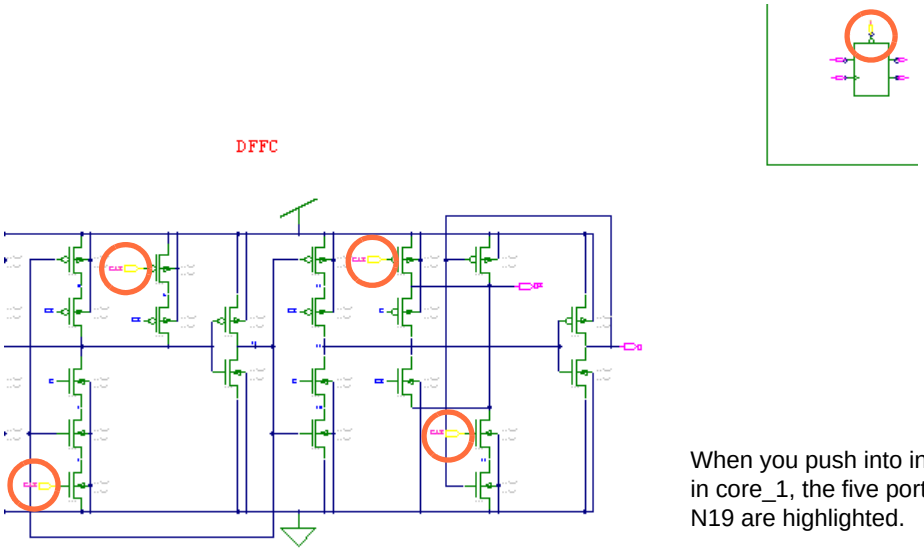
Similarly, if you select a wire or port while pushed into an instance, the net it is connected to will remain selected as you pop back up the design hierarchy.



A wire connecting PadInC_2 to core_1, belonging to net N19, is selected.



When you double-click to push into instance core_1, all of net N19 is selected.



When you push into instance DDFC_3 in core_1, the five ports (circled) on net N19 are highlighted.

6 Creating a Symbol

The process of schematic capture first involves the creation of symbols. Symbols are a pictorial representation of an electrical component together with a definition of the electrical connections that can be made to that component. Symbols may also contain properties that define the electrical characteristics of the component.

In many design situations, a library of symbols already exists, usually of basic standardized components which a designer can use to create their schematic.

Elements of a Symbol View

A symbol view contains the following elements:

Symbol Graphics

Symbol graphics comprise the graphical image of the symbol. This is the image that is seen when the symbol is instanced. Symbol graphics can be boxes, polygons, paths, or circles.

Labels

Text labels can be added to a symbol, and are visible when the symbol is instanced.

Ports

Ports define the connection points that can be made to the symbol when the symbol is instanced. When instanced, ports do not show their text, but appear as an open circle connection point for a wire.

Properties

Properties are name-value pairs that are usually used to describe some characteristic of a device, such as a transistor length, width, or Source/Drain areas and perimeters. Properties can also be used for other purposes, such as controlling the SPICE statements written for a device. Properties on a symbol provide the default values when that symbol is instanced, but may be overridden on an instance-by-instance basis.

How To Create a Symbol

Follow these steps to create a new symbol:

[1] **Create a View**—Create a new cell with a new symbol view, or create a new symbol view of an existing cell using **Cell > New View**. Give the symbol a **View name**, and select the interface it is to be associated with. (See "Creating a New View (Cell > New View or "N")" on page 71.)

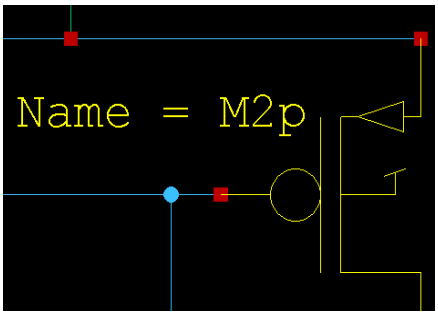
[2] **Draw and label graphics**—Draw a graphical representation of the symbol using the object drawing tools. You can add text with the Label drawing tool. The graphical representation has no electrical meaning, but provides a recognizable way to identify instances of the cell. (See "Drawing Tools for Annotation Graphics" on page 100.)

[3] **Add ports**— Ports define the connection points that can be made to a symbol when the symbol is instanced. They can be of type **In**, **Out**, **In/Out**, **Other** or **Global**. (See "Adding Ports" on page 122.)

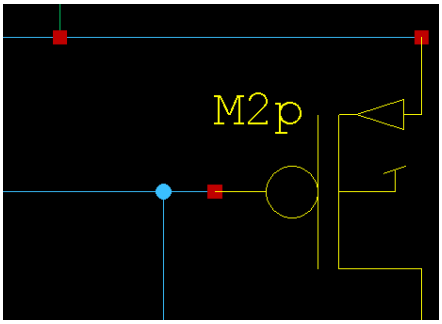
[4] **Add Properties**—If needed, add properties to the symbol. Properties on a symbol provide the default values when that symbol is instanced, but may be overridden on a per instance basis. (See "Symbol Property Types," below.)

Visible and Hidden Properties

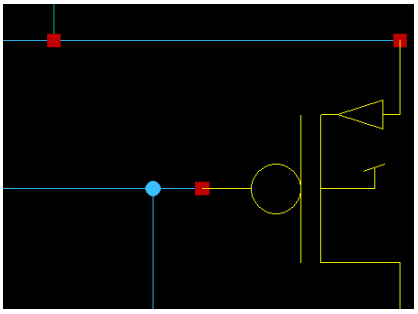
When a symbol is instanced, you can choose to make its properties *visible*, *hidden*, or *value only* in the design area..



"Visible" shows the property name and its value.

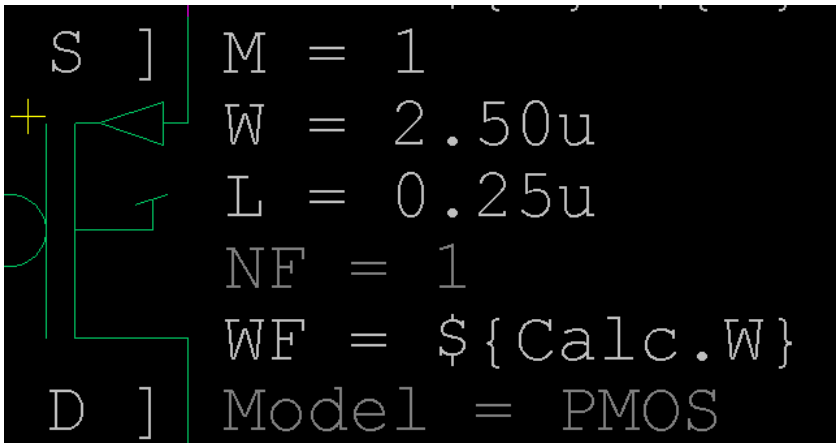


"ValueOnly" shows just the property value.



"Hidden" shows neither the property name or property value.

In the work area, hidden properties are distinguished with a default text color that is lighter gray than the default color for visible and value only properties.



NF and **Model** are hidden properties, **M**, **W**, **L** and **WF** are visible.

Creating and Updating Symbols Automatically

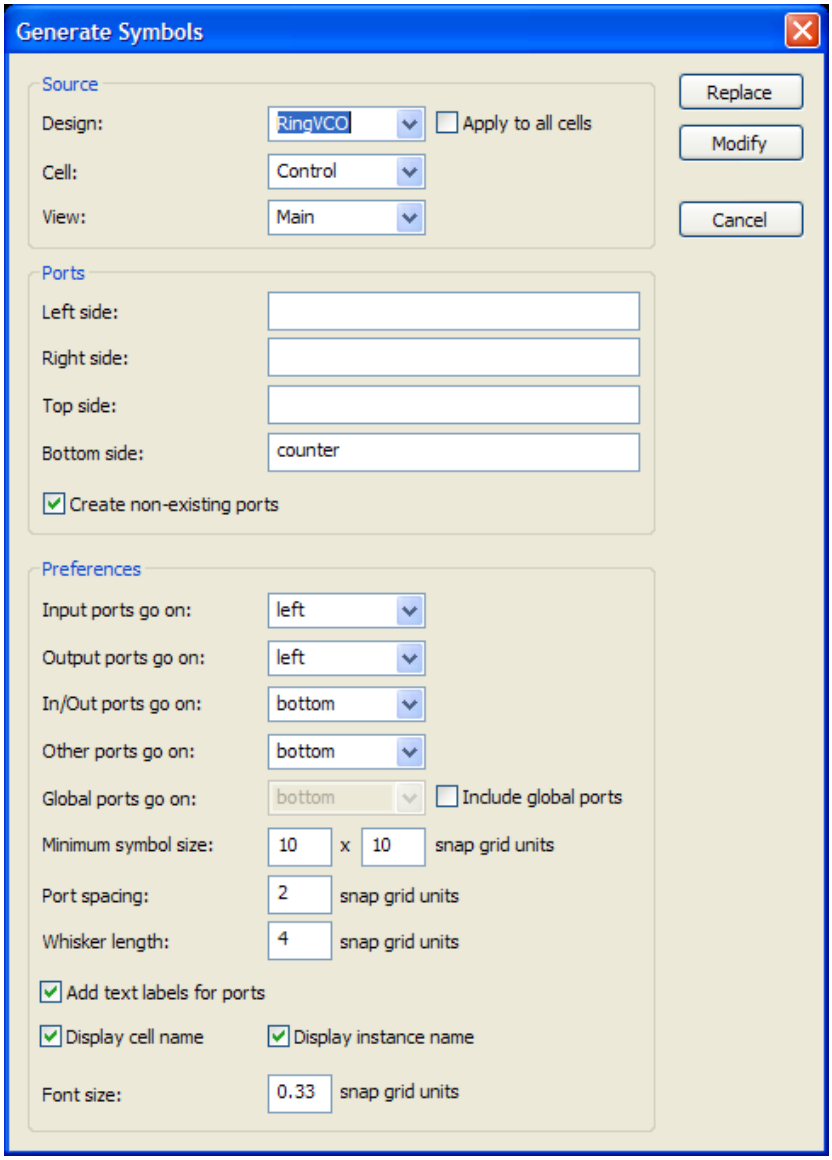
S-Edit can automatically create a symbol from the schematic view (or from an interface view if a schematic view is absent) using **Cell > Generate Symbol**. If the symbol view is empty, **Generate Symbol** reads from the schematic to create a rectangle with ports corresponding to those on the schematic, plus any others you choose to add.

You can specify the side on which ports of different types are placed, the port size, port spacing and whicker length. You can also update existing symbols with new ports added to the schematic. Note that no port name checking is performed.

The snap grid setting is used to ensure that all ports are placed on-grid.

If a port is removed as a result of the **Modify** operation, the whisker will remain, and S-Edit will note the port name and the date and time it was deleted in the work area below the symbol graphics.

Note that when a port is added using **Modify**, it is drawn in the work area below the symbol geometry, unattached to any existing geometry.



Replace Draws or redraws ports on the symbol geometry. Preexisting geometry is erased.

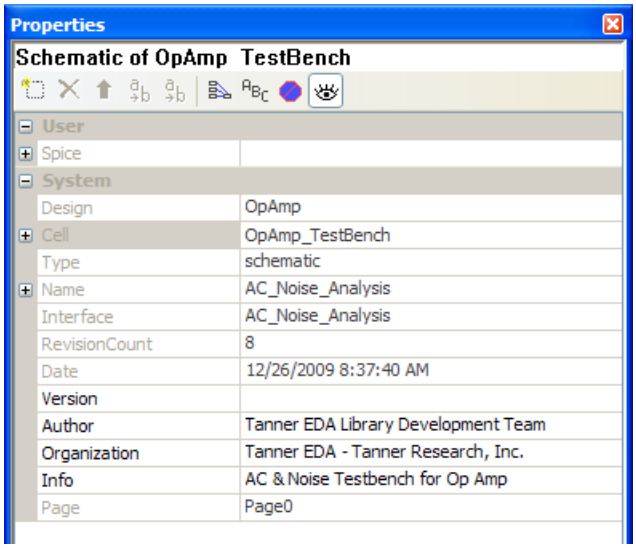
Modify Draws or redraws ports in the work area but not connected to the symbol. Preexisting geometry remains. Deleted or added ports are annotated with name and date/time stamp.

Source Choose a **Design**, **Cell** or **View** for which a symbol will be generated. To generate symbols for an entire design, check **Apply to all cells**.

Ports	Use this section to specify where ports of a given name are drawn, by typing the port name in the **Left side**, **Right side**, **Top side** or **Bottom side** field. Use a space to separate multiple entries. Use an asterisk to place a gap between two ports, where the gap will have the same size as the **Port spacing**. When **Create non-existing ports** is checked, S-Edit will create ports of the name entered, of type In/Out, although they are not found in the schematic.
Preferences	Use the pull-down fields **left**, **right**, **top** or **bottom** to specify the location of ports, by type, on the symbol rectangle.
Input ports go on **Output ports go on** **In/Out ports go on** **Other ports go on**	When **Include global ports** is enabled you can also specify where to locate global ports, using the **Input ports go on** field.
Minimum symbol size	Enter a value for the X and Y dimensions, respectively, of the symbol geometry.
Port spacing	Enter the distance by which each port will be separated.
Whisker length	Enter the length of the line that extends from the symbol rectangle to a port symbol.
Add text label for ports	Check this box to display the port name.
Display cell name	Check this box to display the cell name.
Display instance name	Check this box to display the instance name.
Font size	Enter the text size in snap grid units.

Symbol Property Types

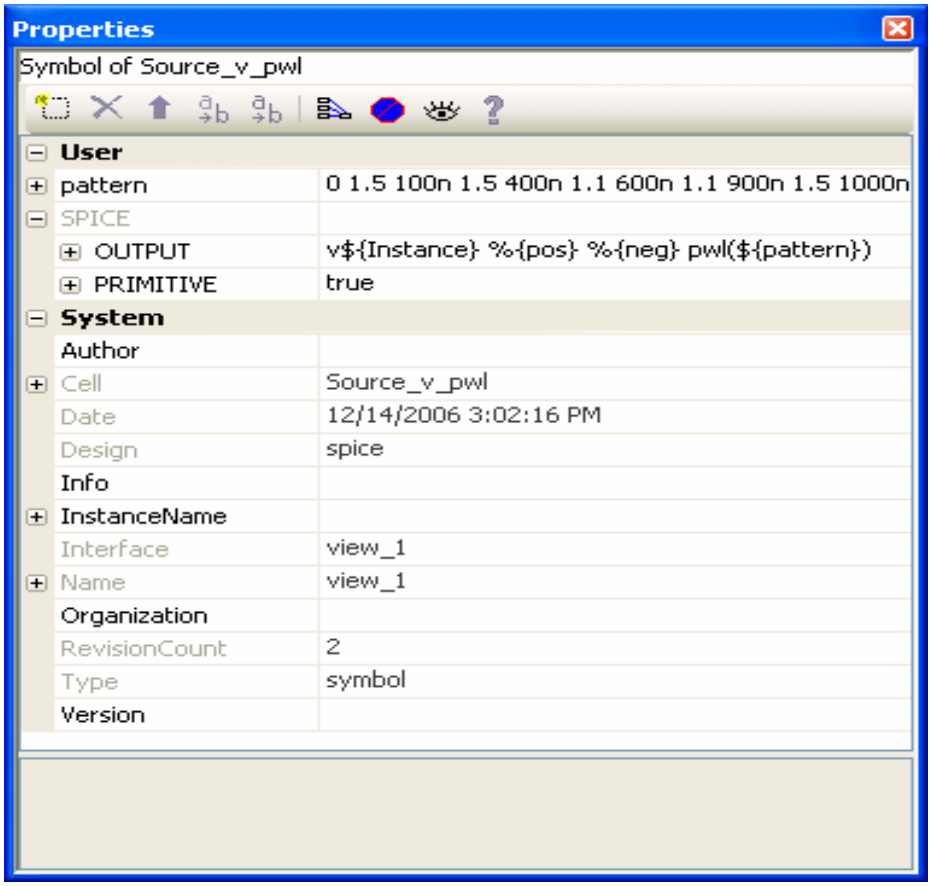
Properties are used to store parametric descriptions of design elements. Properties can characterize a cell's physical parameters, such as length, width, and perimeter; its nonphysical parameters, such as device type and comments; and its output strings, which specify how S-Edit will write a cell to a netlist.



Properties are name-value pairs.

They can be text or a numerical value, or they can be expressions that require evaluation, in particular during instancing or exporting (as shown in the SPICE properties below).

S-Edit allows for properties with no value; a single space " " is read as empty.



System Properties

Properties such as the name of the design or library from which a symbol comes or the cell version are shown in the **System** area of the Properties navigator. In general, the system properties should not be edited.

User Properties

User properties such as Drain Area (AD), Source Area (AS), Length (L) etc. are shown in the **User** area of the Properties navigator.

S-Edit uses property information in two ways. When you instance a cell with properties, S-Edit interprets the properties (as long as they are not hidden) and displays their interpreted values in the instancing cell. If a property has a numeric value, S-Edit displays the value of the number. If a property has a string value, S-Edit displays literal text and the interpreted value of expressions, if any are present (see “Evaluated Properties and Labels” on page 160.)

When you export a netlist, S-Edit writes out the values of output properties appropriate to the netlist format. You can use SPICE output properties to include simulation commands in the netlist. For more information on output properties, see “Exporting SPICE Files” on page 180.

You can define, add, edit and import your own properties for design elements, which may be absolute values or expressions for evaluation.

Default Properties

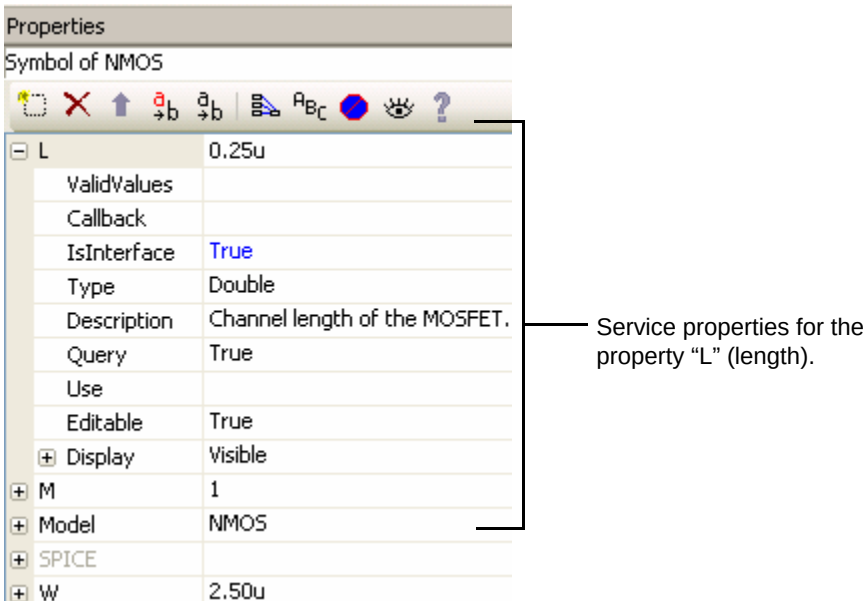
The properties on a symbol are default properties that are used when the symbol is instanced. When you change a property in the symbol view of a cell, that property will change in all the instances of that cell.

Properties for a symbol are visible on the symbol itself as well as in the Properties navigator. The Properties navigator will contain different fields depending on the type of object selected.

You can add, delete, or edit properties at any time, and you can override properties assigned in the original cell with properties assigned to the instance. You can also show or hide a property’s name or value, for most but not all properties.

“Service” Sub-Properties

S-Edit properties carry sub-properties, called *service properties* that are used to set functional characteristics such as allowable values, display characteristics or callback properties.



Valid Values

ValidValues is an enumerated list from which the user can choose values for the property. Setting ValidValues to A B C D will make the value field of the property become a drop-down menu with choices A, B, C and D. To place spaces in a value, surround the value with braces.

Callback

Callback is the name of a TCL function to call when the value to the property is changed. These TCL functions are user-created.

Typical uses of callbacks are to perform validity checking of the input, or to modify other properties that should change in order to maintain consistency with the modified property.

IsInterface

Only properties with "**IsInterface = True**" will get written to SPICE as parameters of a device or subcircuit when $$ is used to write all parameters.

Type

The property type, from the following pull-down choices:

- **String**- text
- **Boolean**- true or false
- **Double**- floating point number
- **Integer**- integer. Properties of this type support expressions (see "Evaluated Properties" on page 151).
- **Coordinate**- position number (may be negative), to be displayed in display units
- **Length**- distance number (non-negative), to be displayed in display units
- **FontSize**- positive number, to be displayed in points

Description

When in Evaluated mode, the property is displayed in the property grid using the description text rather than the variable name. The description text is displayed in the description area at the bottom of the property grid when the property is selected.

Query

Sets display of the property in the property navigator. Only properties whose Query value evaluates to True will be displayed in the property navigator, when in Evaluated mode and the Show Query button in the property navigator is pressed.

Use

Indicates that the property is used in the design.

Editable

Sets the editability of the property in the property navigator. Only properties on instances whose Editable value evaluates to True will be editable in the properties navigator when Evaluated mode is on.Non-Editable properties will be displayed with a gray background when Evaluated mode is on. Properties on symbol views may be edited regardles of the value of Editable.

Display

Sets how a property is displayed when the symbol is instanced. Options are:

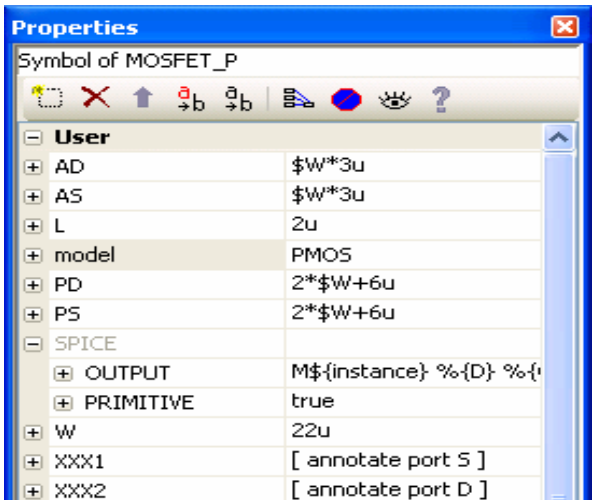
- Hidden—the property is not displayed.
- Visible—displays both the property name and its value.
- ValueOnly—displays just the property value.

Font size, position, justification and orientation are subsets of the display property, as well as WhenNotEvaluated.

WhenNotEvaluated—Sets how an evaluated property is displayed when it has not been calculated. Options are Hidden, Visible and ValueOnly.

Evaluated Properties

A property can be an explicit value or an *expression*.



For example, in the MOSFET_P symbol shown above, the value of **L** is 2u and the value of **W** is 22u. Properties **AD**, **AS**, **PD** and **PS** are expressions which reference the values of other properties using the "$." The drain area (**AD**) is 3u times the gate width (**W**), which is expressed in the Properties navigator as **AD** = $W*3u. This will be evaluated to yield **AD** = 22u*3u = 66p. See the chapter **"Evaluated Properties and Labels**" (page 160) for more detail.

Editing User Properties—the Properties Navigator

You can use the Properties navigator to enter or edit any attribute of a user property (e.g. visibility, font size, or text positioning) on any number of properties that are selected in the active view.

Warning: It is important to note that edits made in the Properties navigator are unrestricted and are not validated by S-Edit.

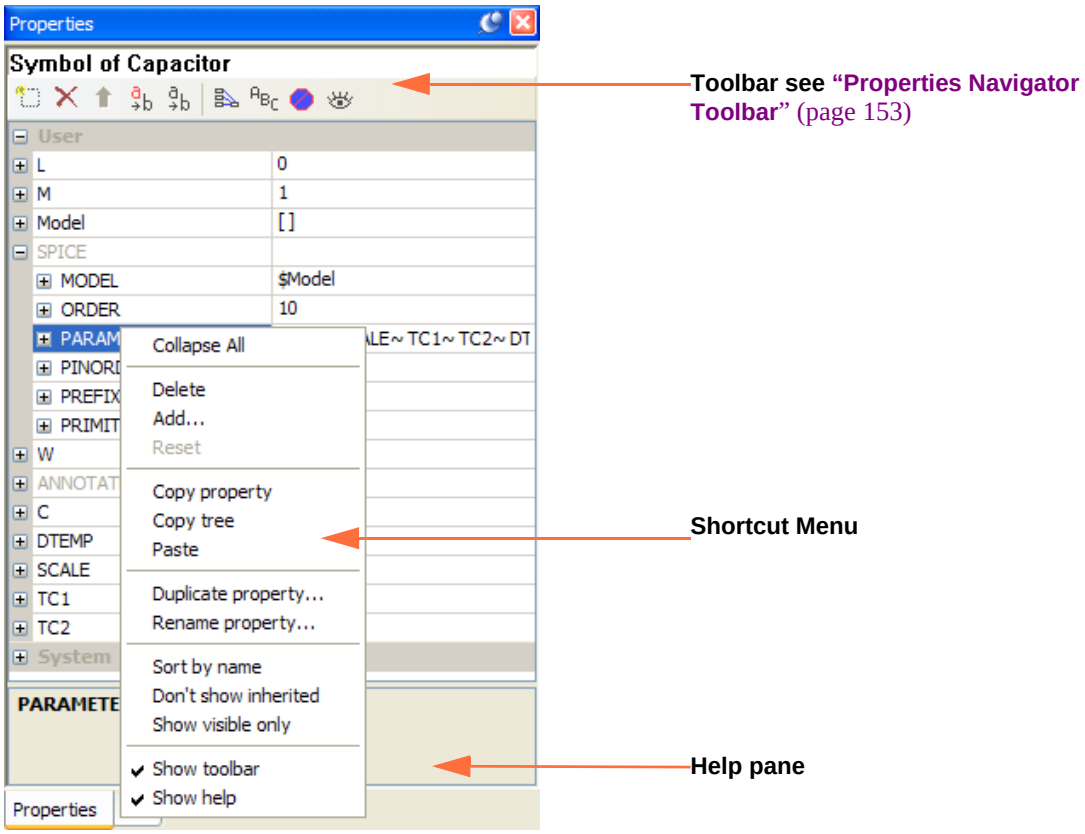
For example, it is possible to draw a 90° line and then convert it to an all-angle line using the value field of the related property. S-Edit does not give a warning if you make such a change.

Expressions that yield evaluated properties are displayed in green in the Properties navigator and cannot be edited.

Note that any edits made in the Properties navigator affect **all** objects that are selected in the layout.

Shortcut Menu for the Properties Navigator

Right-click anywhere in the Properties navigator to open the shortcut menu shown below.

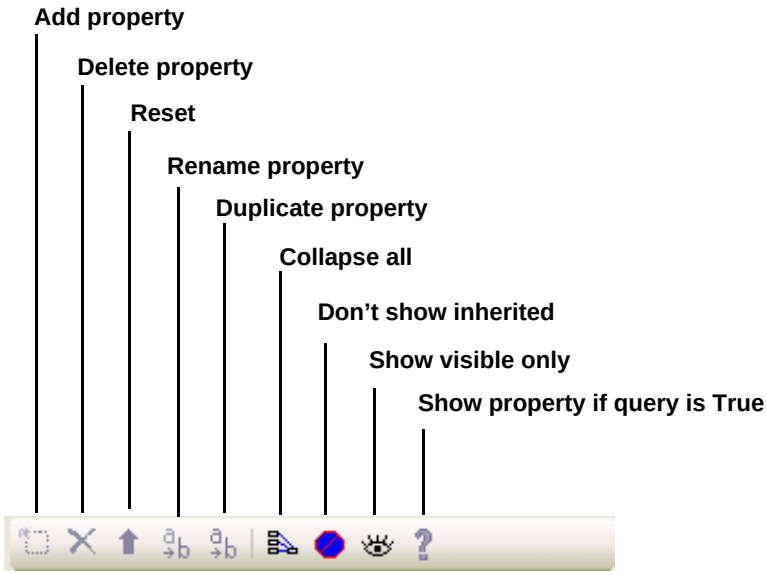


Collapse All	Collapses all properties at the highlighted level of the hierarchy.
Delete	Deletes the highlighted property. Use **undo** to restore.
Add	Opens a dialog so you can add a user property.
Reset	Resets an instance property value to that of its source cell.
Copy property	Copies data for a property and its parent properties (one level up the hierarchy) in **.tcl** file format.
Copy tree	Copies the entire property hierarchy (above and below the selected property) in **.tcl** format.
Paste	Pastes copied property contents to the command window in **.tcl** file format.
Duplicate Property	Opens a dialog where you can duplicate the highlighted property.
Rename Property	Opens a dialog where you can enter a new name for the highlighted property.
Sort by name	Sorts properties alphabetically, separately by category.
Don't show inherited	Hides properties inherited from the symbol primitive.
Show visible only	Hides property values that are set to hidden in the design.

Show Toolbar	Toggles the Properties toolbar display on (checked) or off (unchecked).
Show help	Toggles the display of property descriptions in the lower pane on (checked) or off (unchecked).

Properties Navigator Toolbar

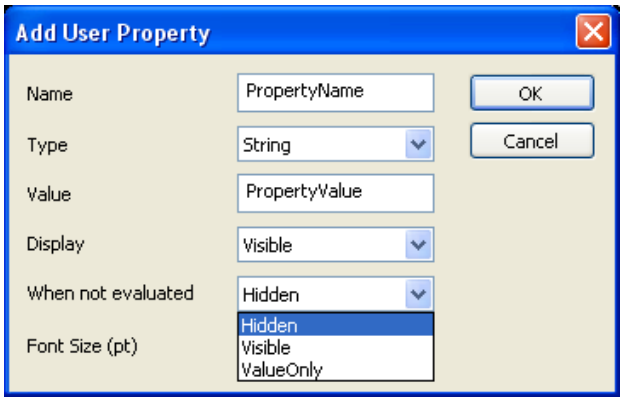
The toolbar buttons give you quick access to the most common property editing operations.



Adding User Properties

User properties may be created, for example, to count cell instances, to reference ports or other properties, or to format property outputs. S-Edit interprets user properties in instances (when shown in schematic mode) or during export to a netlist. S-Edit parses all other text without expansion.

You can add user-defined properties with the **Add Property** button in the Properties navigator, the **Property** icon , or with the command **Draw > Electrical > Property**. S-Edit opens the **Add User Property** dialog when you click in the work area after any of these commands.



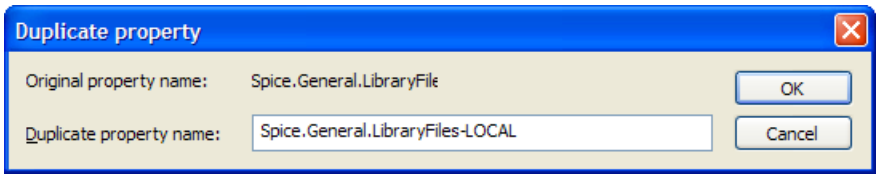
Duplicating and Renaming Properties

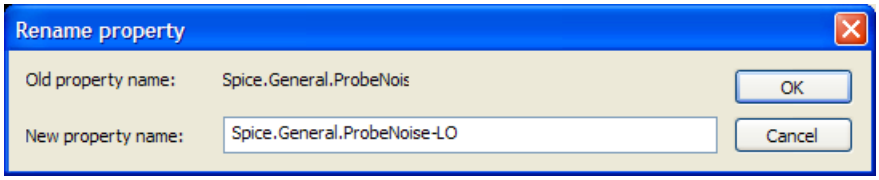
Properties can be renamed or duplicated by right-clicking on the property in the Properties browser and selecting **Rename Property** or **Duplicate property** respectively.

Renaming a property on an instance in schematic creates a duplicate of the property with the new name, as the original in the symbol must not be modified. When you choose **Duplicate property,** S-Edit duplicates all data for the highlighted property and opens a dialog where you can enter a name for the new property. Similarly, for **Rename property** S-Edit opens a dialog that lets you give the highlighted property a new name.

You cannot rename a property if it is a system property, the property is a group or service rather than a value, or the property is inherited.

When you click **OK** S-Edit enters the updated property in the navigator.





Copying and Pasting Properties

The **Copy** and **Paste** operations issue TCL commands to the S-Edit Command window so that user properties and properties of views can be transferred from one object to another.

These operations are object-independent—the source and target objects can be of different types, and can be anything that has a property attached. This is particularly useful for saving and reloading complex properties like SPICE simulation settings.

Note that if a property of the same name exists in the paste destination it will be overwritten without a warning.

You have the option to select a property and all values below it (descendents) using **Copy tree**, or just a single property using **Copy property**.

Copy saves and converts properties to TCL format commands which are written to the clipboard. **Paste** issues a background source command that loads the TCL commands from the clipboard directly into the S-Edit Command window. You can also paste and view the copied TCL file in any text editor.

Replacing a Device or Symbol Globally

You can effectively replace a device or symbol with another by changing the **MasterCell** property in the Properties navigator. When you do so S-Edit will globally replace all instances and also indicate the change by appending **_Unresolved** to the name of the library as it appears in the Libraries list.

Selecting Properties from the Layout

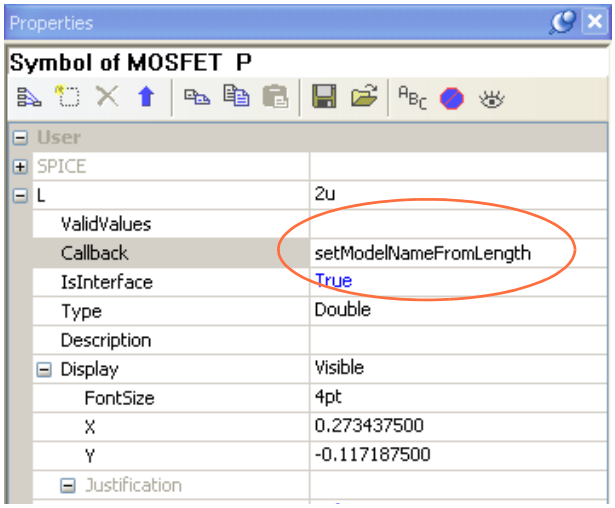
From a design window, use **Ctrl + click** to select a property and make it active in the Properties navigator.

You can also **right-click** on one property in the layout (it will not appear to be selected) then use **Ctrl + right-drag** to select multiple properties (selection boxes will be visible at this point).

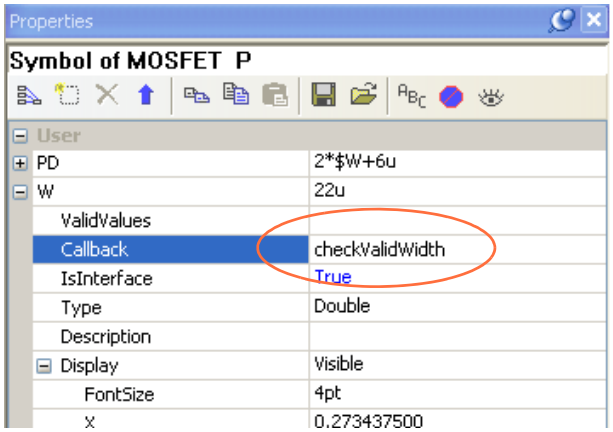
"Callbacks" for Property Values

Callbacks provide the ability to call a command upon changing a property value. The callback command is usually a user-written function. Typical uses of callbacks are to perform validity checking of the input, or to modify other properties that should change in order to maintain consistency with the modified property.

For example, on the symbol view of a cell **MOSFET_P** there is a callback function for property **L** named **setModelNameFromLength**. This function retrieves the length value entered and sets the model name based on that length.



Similarly, expanding property **W** shows that it has a callback function named **checkValidWidth**, which checks that the width you entered is in the range 0.25u = W = 50u, and returns an error message and disallows the value if it is not.



If you were to open the schematic view of a cell containing an instance of **MOSFET_P** and change the value of **L** from 1u to 0.1u, the value of model **PMOS.3** with L=1u will automatically change to **PMOS.1** due to the callback.

If you change **L** back to 1u, the model value will change back to **PMOS.3**. However, if you try to change the value of **W** to 0.1u, S-Edit will display a callback error message in the log window stating that the value is too small.

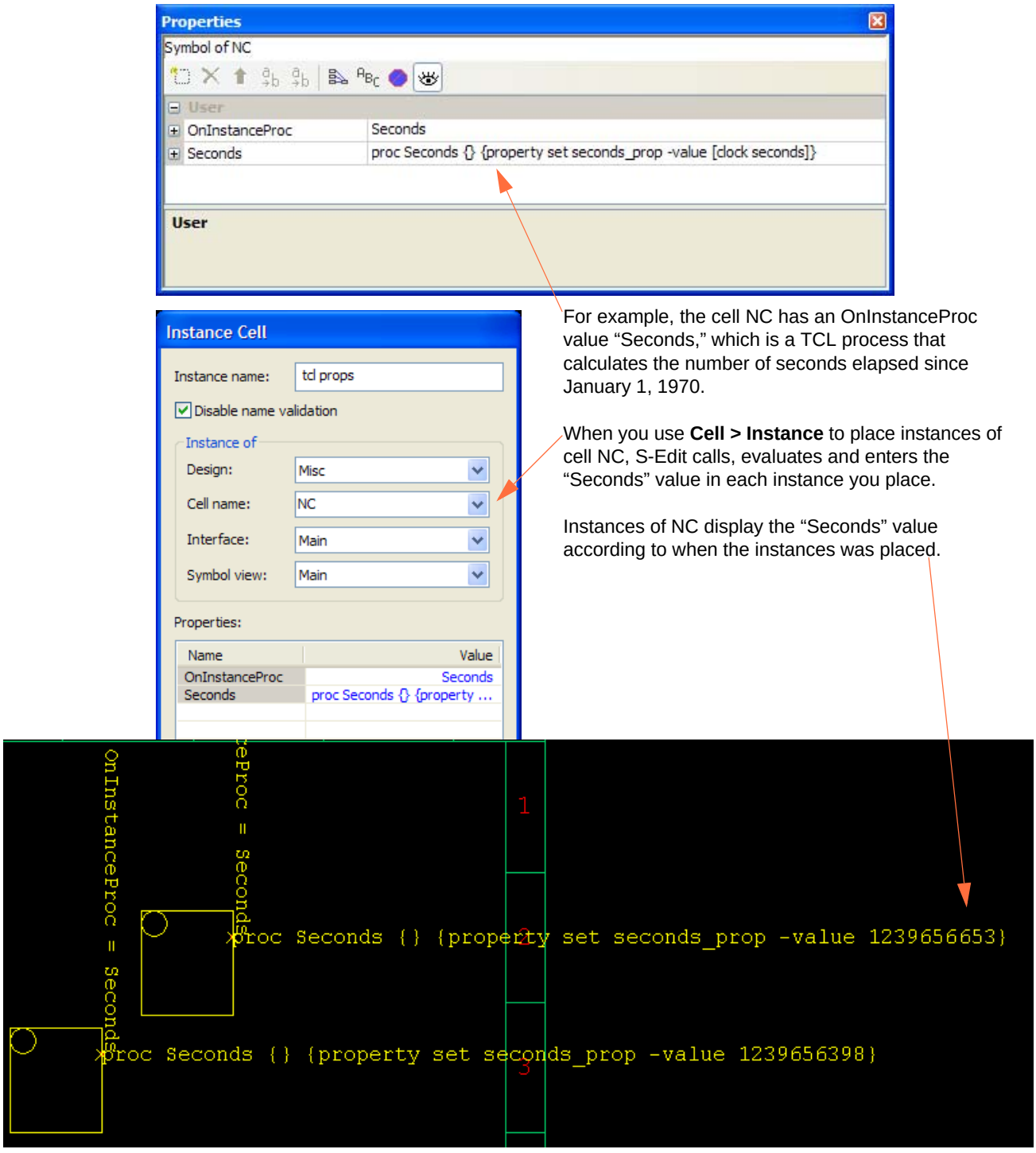
S-Edit can import callback function names and parameters from Cadence EDIF so that schematic views include this information in the proper location. When this capability is enabled, instead of having to find and populate each callback property field, S-Edit creates a list of the callback procs along with some basic “fill in the blank” formatting. You then enter the function definitions in these “callback stubs.”

Automatically Setting Properties During Cell > Instance

You can automatically change or update a property value with the special user property “**OnInstanceProc**.” The value in the **OnInstanceProc** field names a TCL process that S-Edit calls when the symbol is instanced. You write that process to set the value of any of the symbol properties you choose. When you use **Cell > Instance** to instance a cell that has an **OnInstanceProc** value, that TCL script named runs and populates the user properties it controls.

OnInstanceProc is particularly valuable when you have a library of generic symbols you want to use with different manufacturing processes. Instead of creating a separate set of cells for each process, you can write scripts that call process-specific variables, and use those scripts in the OnInstanceProc field.

The appropriate properties are automatically entered or updated as you place instances of the cells with an OnInstanceProc defined.



For example, the cell NC has an OnInstanceProc value “Seconds,” which is a TCL process that calculates the number of seconds elapsed since January 1, 1970.

When you use **Cell > Instance** to place instances of cell NC, S-Edit calls, evaluates and enters the “Seconds” value in each instance you place.

Instances of NC display the “Seconds” value according to when the instances was placed.

Writing TCL Functions for Callbacks

The user-written functions **setModelNameFromLength** and **checkValidWidth** are shown below:

```
proc setModelNameFromLength {} {
       set cellname [ property get MasterCell -system ]
       if  {[string equal $cellname "MOSFET_P" ] == 1 } {
```

```
            set basemodel PMOS
          } else {
            set basemodel NMOS
          }
      set len [ stod [ property get L ] ]
      if { $len < 0.25e-6 } {
            property set model -value ${basemodel}1
            return
      }
      if { $len < 1e-6 } {
            property set model -value ${basemodel}2
            return
      }
      property set model -value ${basemodel}3
}

proc checkValidWidth {} {
      set width [ stod [ property get W ] ]
      if { $width < 0.25e-6 } { return "width too small"}
      if { $width > 50e-6 } { return "width too large"}
```

The callback functions must be defined in S-Edit before they can be called. This is done either by dragging the file containing the callbacks into the command window, or by placing the file in one of the folder locations from which scripts get automatically loaded.

Scripts placed in a folder **scripts/open.design** in the design folder will be automatically loaded when the design is opened. The other locations and conditions by which S-Edit automatically loads scripts are shown in the following table.

To load script when any design is opened, place script in:	C:/Documents and Settings/<*username*>/Application Data/Tanner EDA/scripts/open.design
To load script when S Edit is started, place script in:	C:/Documents and Settings/<*username*>/Application Data/Tanner EDA/scripts/startup
To load script when S Edit is shutdown, place script in:	C:/Documents and Settings/<*username*>/Application Data/Tanner EDA/scripts/shutdown

Port Placement

S-Edit is more flexible than many other schematic editors with respect to port placement. S-Edit allows you to place a port at any location on the symbol, whereas other schematic editors require that you place the port on the symbol boundary. If you plan to export your design to another schematic editor, you will probably achieve better results by following this convention and placing your ports on the symbol boundary.

When you instance a cell, S-Edit translates the ports on the symbol view into ports on the cell instance. Ports thus provide connection points between lower-level cells and their instances. When you connect objects to a port on a higher-level instance, you also connect them to any object connected to the corresponding port on the schematic of the originating cell.

For example, when you connect an object to the input port **In** of an instancing inverter, (corresponding to a port named **In** on the symbol view of the inverter cell), you have connected it to the net named **In** on the schematic of the instanced inverter cell.

7 Evaluated Properties and Labels

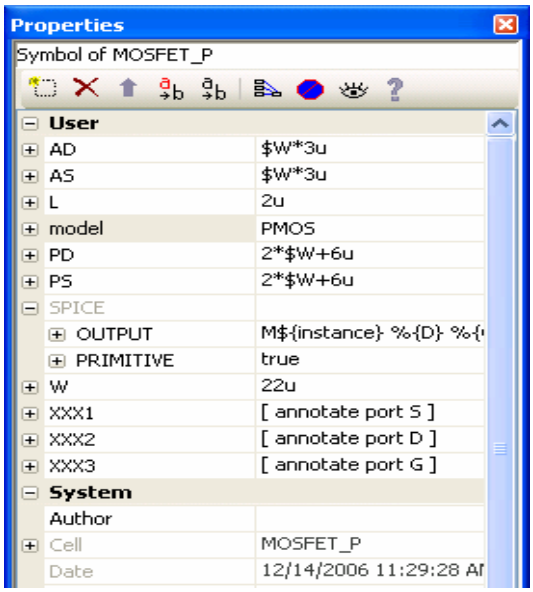
S-Edit supports expressions as property values; in fact, every property value is implicitly an expression.

Expressions as Property Values

A property can be an explicit value, or an expression which references the values of other properties.

For example, in the MOSFET_P example below, the value of **L** is 2u and the value of **W** is 22u. Properties **AD**, **AS**, **PD** and **PS** are expressions which reference the values of other properties using the "$."

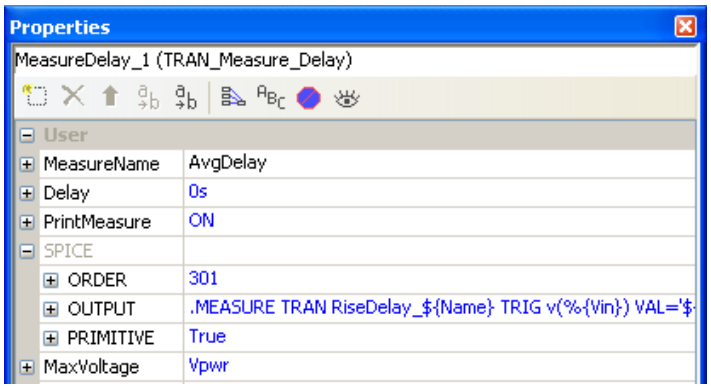
When used on a symbol, the "$" references a value on the symbol, and when used on an instance, the "$" references a value on the same instance. In this symbol, the Drain area (**AD**) is 3u times the gate Width (**W**), which we have expressed in the property browser as **AD** = $W*3u. This will be evaluated to yield **AD** = 22u*3u = 66p.



Displaying Evaluated Properties

Properties can be displayed in the property navigator either as their original expression, or as the evaluated result of their expression. To display the evaluated result of properties you must turn on

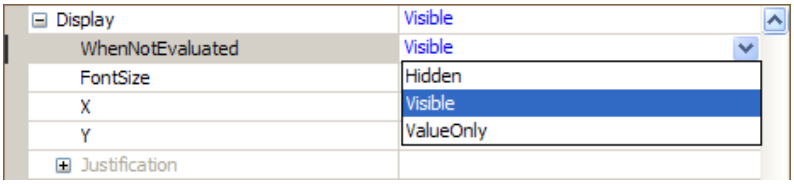
"Display Evaluated Properties". Properties can be edited in either display mode. A property will display it's expression when being edited in Evaluated mode..



To display evaluated values of properties, turn on "**Display Evaluated Properties**" using the P icon in the Spice Simulation toolbar (as shown below.)

How a property is displayed when a symbol is instanced can be set by setting the **Display** subproperty to "**Hidden**," or "**Visible**," which includes the property name and its value, or to"**ValueOnly**." If the property is written as an expression, you can use the **WhenNotEvaluated** field to apply these three visibility options when the property is not evaluated.

A property may be made read-only in Evaluated mode by setting the **Editable** sub-property to **False.** The Editable sub-property can also be an expression, allowing one to dynamically control the editablility of a property.. The value of non-Editable properties will be displayed with a gray background



Evaluated Text Labels

S-Edit supports TCL expressions in text labels. A property on a text label called Evaluation, with value TRUE, FALSE or EXPRESSION, indicates whether the the label is to be evaluated. Here, EXPRESSION is an expression that evaluates to TRUE or FALSE.

Supported Operators, Functions and References

S-Edit expressions support standard mathematical operators -, *, /, **, as well as standard functions such as sin(), cos(), etc.

The prefixes %, $, ?, and @ on a property value are used to reference other properties. Braces are optional, and should be used when the property name has a space, or when abutting something immediately after property name.

Expressions can also reference the vales of other properties using the following prefix operators.

Reference	***Interpretation***
%port or %{port}	References the name of the node connected by the terminal T. In the "in context" view, this name is the hierarchical name of the net.

Reference	*Interpretation*
$P or ${P}	Refers explicitly to another property on the same instance, or in symbol view to another property on the symbol. For example, in the expression AD=$W*6u, the $W refers to the value of a property W on the same device. The reference for a property is first looked for as a User property, and then as a System property. If $P does not exist as either a user or a system sibling property, $P is looked for as a TCL variable. This option corresponds to the Cadence **iPar()** function.
?P or ?{P}	References the parent cell's property, but only looks one level up in the hierarchy. As usual, overridden properties on the instance of the parent have higher priority than the default value on the symbol. In particular, a property value **?P** on an instance looks for the property P first on the instance of the parent cell, then on the schematic page containing the instance, then on the symbol of that schematic, and then for a TCL variable. As with the **$** reference, at each level, the property is first looked for as a User property, and then as a System property. (continued)
?P or ?{P} (continued)	Note that in SPICE export mode, parent properties are not expanded as they were prior to the version 12 release, unless: 1) referring to an instance property, 2) the "?" syntax is used, 3) the property exists on the parent view and 4) if the property on the parent view is a system property. For example, consider a symbol of cell BOT instanced in a schematic of cell MID. S-Edit will resolve a property with value **?P** on an instance of BOT by looking for a property P as follows: 6 Look for value of property P on the instance of cell MID. This assumes you have navigated in context through an instance of MID and are now looking at a symbol of BOT, or, similarly, are exporting SPICE from a higher level schematic. (The value on an instance of MID is first looked for as an override on the particular instance of MID, and then on the symbol of MID.) 7 Look for a value of property P on the schematic of MID. 8 Look for a value of property P on the symbol of cell MID. (This is not needed when BOT is looked at in a particular context of MID, but provides a default value when MID is opened without context, or as a toplevel cell) 9 Look for a value of property P defined as a TCl variable. The **?** can be used to iteratively look up the hierarchy. In the above example, cell MID can be instanced in a schematic of cell TOP. The value of property P on the instance of cell MID can be ?Q, and Q could have its value defined on an instance of TOP. This option corresponds to the Cadence **pPar()** function.

Reference	*Interpretation*
@P or @{P}	References the highest-level definition of P. If we have a cell TOP which contains an instance of MID, which contains an instance of BOT, then the priority order of a property P "inside" BOT is then (high to low): global, MID_{INST}.P, MID_{SYM}.P, BOT_{INST}.P, BOT_{SYM}.P.

TCL Commands in Expressions

In the expression for a property value, a string may be passed to the TCL interpreter for evaluation. Any substring in a property value contained in brackets **[ ]** is passed to the TCL interpreter; if the result is successful, it is included verbatim, otherwise the TCL error string is included.

For example, consider that the following TCL process is defined in S-Edit:

```
proc size { val } {
   if {[string compare $val small ]== 0} {return 5}
   if {[string compare $val large ]== 0} {return 20}
   return 10
}
```

An instance (or a symbol) of a cell could have a property A = [size $B]. If there is a property B = small on that instance, then property A after substitution of B becomes A = [size small], and after TCL evaluation becomes A = 5. Similarly, one could have A = [size ?B] or A = size @B].

Built in TCL functions

S-Edit makes use of some of the built-in TCL functions to evaluate expressions.

Selective Evaluation: se

S-Edit uses the built-in TCL command "se" to selectively evaluate properties and format the result depending on whether the property is defined or not.

The syntax of the command is:

`se` ***property true-clause { false-clause }***

If the property exists, the true-clause is returned; otherwise, the (optional) false-clause is returned. Because property substitution precedes TCL evaluation in property evaluation, the se function is needed to identify the case where a nonexistent property returns an empty string.

For example:

```
[se $L {L=$L} {L=2u}]
```

will either evaluate to L=2u or L=<the value of the property L>.

Annotate Commands

Built in “annotate “ TCL commands can be used to annotate various values on the schematic, including values on ports, instance parameters, small signal operating point parameters, model parameters, and device parameters.

The TCL**annotate** command supports the following subcommands:

annotate port <PortName>

annotate instance <InstancePropertyName | InstanceLabelSetIndex >

annotate op <SmallSignalName | OpLabelSetIndex>

annotate model <ModelParameterName | ModelLabelSetIndex>

annotate device <DeviceParameterName | DeviceLabelSetIndex>

annotate param <name | ParamLabelSetIndex>

and flags:

-name	returns the name of he parameter, rather then the value
-digits n	Specifies the number of significant digits to present the result, when it’s a number.
-suffix “string”	Adds the string to the end of the formatted result, in the case when the formatted result is not empty. This allows constructs such as “[annotate param 1 -name -suffix =][annotate param 1]”. This avoids a “=” by itself displayed on the screen if the “=” were placed outside the [], when the Annotate Param dropdown is set to None.

The InstanceLabelSetIndex, OpLabelSetIndex, ModelLabelSetIndex, and DeviceLabelSetIndex options refer to the index in the following system properties on the symbol

InstanceLabelSet	List of Instance properties to be displayed by InstanceLabelSetIndex. InstanceLabelSetIndex = 2 will display the second property in the InstanceLabelSet list of properties.
OpLabelSet	List of T-Spice Small Signal Operating Point parameters to be displayed by OpLabelSetIndex. OpLabelSetIndex = 2 will display the second parameter in the OpLabelSet list of parameters.
ModelLabelSet	List of T-SPice Model parameters to be displayed by ModelLabelSetIndex. ModelLabelSetIndex = 2 will display the second parameter in the ModelLabelSet list of parameters.
DeviceLabelSet	List of T-SPice Device parameters to be displayed by DeviceLabelSetIndex. DeviceLabelSetIndex = 2 will display the second parameter in the DeviceLabelSet list of parameters.

The display of the **param** option is controlled by the **Annotate Param** drop down in the Spice Simulation toolbar. The drop-down lists the following options:

Instance Params

OP Results

ModelParams

Device Params

None

The InstanceLabelSet, OpLabelSet, ModelLabelSet, and DeviceLabelSet system properties should be created on the symbol with the following command:

```
property set -name <PropertyName>-system -host view -value {<list of property values}
```

For example

```
property set -name InstanceLabelSet -system -host view -value {-Model Annotate Port:
```

You can use the built in TCL command **annotate port** to display certain values of interest on the ports of an instance. These include:

- Port name
- Net name
- DC Voltage
- DC Current
- DC Charge

The first two of these are always available to display, the next three depend on values being present from a DC simulation run. To annotate a value associated with a port T, place an evaluated label near the port

```
[annotate port T]
```

You can set which of the annotated property values to display in the work area by selecting from the drop-down menu under the **Display Evaluated Properties** button on the Spice Simulation toolbar. The default state is **Visible**, and all other display characteristics such as font size and justification are saved and reloaded whenever you launch S-Edit.

When displaying annotated values, S-Edit uses the first operating point values in the simulation.

If annotation values are not visible it may be that the Display service property “WhenNotEvaluated” is set to **Hidden** or **value only** (see “Service” Sub-Properties on page 149).

Annotate Instance:

To annotate an instance property “L”on schematic, one can use several methods:

Method 1:

Place an evaluated text label on the symbol directly referencing the property you wish to display

L = [annotate instance L]

Method 2

Place an evaluated text label on the symbol referencing an index into the system property InstanceLabelSet. Here “L” is the third value in the InstanceLabelSet list.

[annotate instance 3 -name] = [annotate instance 3]

Method 3:

Place an evaluated text label on the symbol dynamically referencing an index into one of the LabelSet system properties. Here “L” is the third value in the InstanceLabelSet list. When the **Annotate Param** drop down in the Spice Simulation toolbar is set to **InstanceParams**, the label will display the annotation for L.

[annotate param 3-name -suffix =] [annotate param 3]

Method 4:

An alternative to using the Param method is to use **cdsParam(n)** labels . A text label “**cdsParam(n)”** will display the nth value in the LabelSet chosen by the **Annotate Param** drop down. cdsParam(n) labels are placed on symbos when importing CDF data from Cadence.

Annotate Op:

To annotate an small signal parameter “vgs”on schematic, one can use several methods:

Method 1:

Place an evaluated text label on the symbol directly referencing the property you wish to display

vgs= [annotate instance vgs]

Method 2

Place an evaluated text label on the symbol referencing an index into the system property OpLabelSet. Here “vgs” is the third value in the OpLabelSet list.

[annotate op3 -name] = [annotate op 3]

Method 3:

Place an evaluated text label on the symbol dynamically referencing an index into one of the LabelSet system properties. Here “vgs” is the third value in the OpLabelSet list. When the **Annotate Param** drop down in the Spice Simulation toolbar is set to **OpParams**, the label will display the annotation for vgs.

[annotate param 3-name -suffix =] [annotate param 3]

Method 4:

An alternative to using the Param method is to use **cdsParam(n)** labels . A text label "**cdsParam(n)"** will display the nth value in the LabelSet chosen by the **Annotate Param** drop down. cdsParam(n) labels are placed on symbos when importing CDF data from Cadence.

Annotate Model, Annotate Device:

Annotation of Model Parameters and Device Parameters follow the same syntax as illustrated above for Instance and Op parameters.

Cadence cdsParam(n) Labels

OpenAccess or EDIF databases written by Cadence may contain **cdsParam(n)** labels on symbols. These are used to annotate values on schematic, similar to the **Annotate Param** method. In Virtuoso, the **n** in **cdsParam(n)** indexes into one of three label sets, paramLabelSet, opPointLabelSet, or modelLabelSet. When a Cadence OA database or CDF file is imported, the paramLabelSet, opPointLabelSet, and modelLabelSet. are mapped into system properties on the symbol as follows.

cdf Parameter	***LabelSet***
paramLabelSet	InstanceLabelSet
opPointLabelSet	OpLabelSet
modelLabelSet	ModelLabelSet

The LabelSet that **cdsParam(n)** references is controlled by the **Annotate Param** drop down. If a parameter name in these lists is preceded by a minus sign (-), then when cds.Param is displayed, the parameter value is displayed, but the parameter name is not.

Viewing Property Values In Context

You can use the **Push into Context** button when an instance is selected to open a specific instance of a cell. Depending on the object and type of analysis, when you push to deeper levels of the design hierarchy you can see, for example, small-signal parameter values, property values derived from expressions, or operating-point voltages.

While thus editing "in context," you can only select or edit objects contained in the instance. However, you can continue to push down to lower levels of instances within an instance until you reach a SPICE primitive.

Use the **Pop Context** icon to "pop out" of the last instance you pushed into until you return to the top level of the cell hierarchy.

Note that if you are "pushed in" to a cell schematic, you can double-click on an open space to move a level up in the circuit hierarchy.

8 Importing and Exporting Netlists and Schematics

Importing a Design

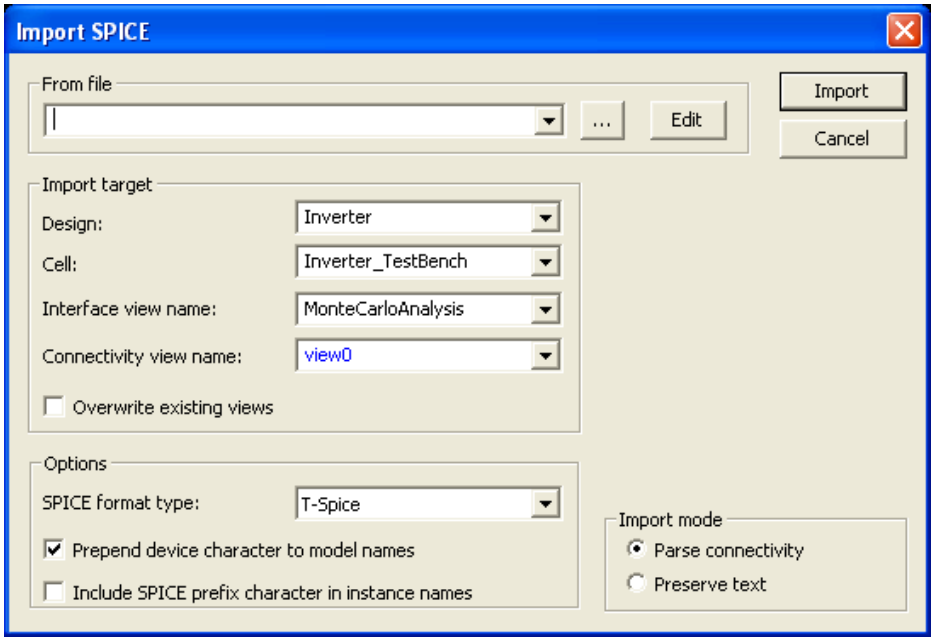
You can import EDIF, SPICE and Verilog files into S-Edit, and export EDIF, SPICE, Verilog and VHDL files from S-Edit. No matter what format you are importing, you must have a design open before you can import. The import operation creates a complete project directory for the design.

Importing SPICE Files

SPICE import creates SPICE views, which are saved with the design. Importing a hierarchical netlist will create separate cells for each subcircuit in the netlist.

When the hierarchy priority results in a spice view being exported, that view is exported verbatim.

Note that the SPICE view is regenerated from the connectivity view, so it is not a verbatim copy of the original netlist. As a result, device parameters are kept but comments and additional commands that are not part of the connectivity will be lost.



From file — Specifies the SPICE file to import.

Edit — Opens the specified file in the S-Edit text editor.

Import target — Specifies the **design**, **cell**, **interface view**, and **connectivity view** into which the SPICE file will be imported.

Overwrite existing views	If cells of the same name exist in the SPICE file and the design it is being imported to, a check in this box causes the SPICE file to overwrite the views in the existing cell.
SPICE format type	Choose from **T-Spice**, **HSPICE**, **PSpice** and **CDL**. Pin type from CDL is imported via the .PININFO command.
Prepend device character to model names	When checked, SPICE device characters will be prepended to model names in the netlist.
Include SPICE prefix characters in instance names	When checked, SPICE prefix characters will be included in model names so that the entire word becomes the instance name rather than stripping the first character.
Parse connectivity	When checked, parses connectivity from a SPICE netlist including files referenced by .include statements. Spice views are then generated from the connectivity views.
Preserve text	When checked, creates a Spice view cell for each subcircuit under the subcircuit name, and pastes in the entire subcircuit text including comments, include files and formatting. Comments written between subcircuit definitions are included in the subcircuit cell below them. Creates “cell0” for the top-level cell.

It is recommended that you check the model files of commercially available device, as they may have critical parameters included with the model parameters rather than in the device parameters as T-Spice expects, or they may include parameters that are not valid for your model level.

Importing EDIF Files

The EDIF import operation creates SPICE views. As with importing a SPICE file, since the views are saved with the design, the imported EDIF is saved. Importing a hierarchical netlist creates separate cells for each subcircuit in the netlist. New libraries created by importing EDIF are opened with the settings editing allowed and with exclusive access.

If you wish to create a new design from an EDIF file, you should create a design with the same name as the top level or root library in the EDIF file, and then import the EDIF file into that design. The root library is often written near the bottom of the EDIF file with a “design ROOT” entity as:

```
(design ROOT
            (cellRef rootcell
                        (libraryRef rootlibrary)))
```

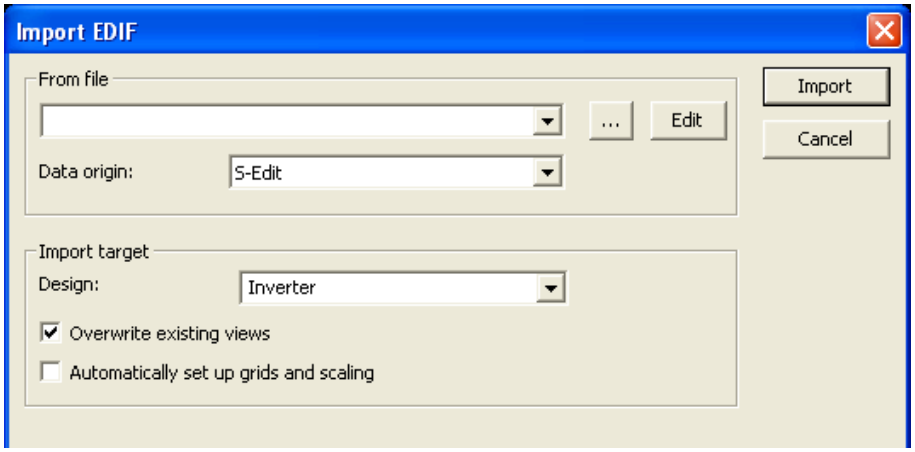
Where *rootlibrary* is the name of the root library. The root library is the one nearest the bottom of the EDIF file, so you can also identify it by locating the entity “library *libraryname*” that is nearest the bottom of the file.

Another way to create a new design from an EDIF file is to create a design with any name, import the EDIF file, and save the design and its libraries. Use the Top-level filter in the Libraries navigator to assist in finding the toplevel library. You can then open the toplevel library directly (in this context it becomes the design) and you can delete from disk the placeholder design that you initially created.

Note that standard cell place and route (SPR) in L-Edit requires either a flattened EDIF netlist or one with only one level of hierarchy.

Importing EDIF from S-Edit or ViewDraw

The import choices will vary depending on the type of EDIF file you are importing. If it is an EDIF file from S-Edit, ViewDraw or an unknown source, translation parameters are not required. Cadence Composer and Mentor Design Architect files each require their own set of options. .



From file	Specifies the EDIF file to import.
Edit	Opens the specified file in the S-Edit text editor.
Data origin	Select the type of EDIF file from the menu.
Import Target	Specifies the design into which the EDIF file will be imported. If a library exists in the EDIF file with the same name as the design or one of the libraries of the design, then the contents of that EDIF library will be imported into the design or library of the same name. Libraries in the EDIF file that do not match the design name will be created as new libraries of the selected design.
Overwrite existing views	If cells of the same name exist in the EDIF file and the design it is being imported to, a check in this box causes the EDIF file to overwrite the views in the existing cell.
Automatically set up grids and scaling	Causes S-Edit to analyze the contents of the EDIF file for pin spacing and calculate the best grid spacing and scaling.

So that nodes will retain their connectivity when they are not explicitly connected with wires, S-Edit adds net labels to named wires (prior to segmentation), at the end of the wire if it is explicitly named.

EDIF Translations for ViewDraw

When S-Edit imports EDIF from ViewDraw it performs these translation:

[1] All instances of a cell called 'SPLITTER' are removed, and cell SPLITTER itself is removed. This is a ripper cell that S-Edit does not need.

[2] Buses and bus components are renamed to adhere to S-Edit syntax. For example, D<0:7> for a bus, D<1> for a single bit from the bus.

[3] The leading '@' is removed from property names.

[4] The leading '$' in instance names is replaced with '_'.

[5] A SPICE.OUTPUT property is created from the ViewDraw properties PREFIX, PINORDER, and ORDER.

[6] A SPICE.PRIMITIVE property is created on a symbol and its value is set to 'true' only if there is a property 'PREFIX' on the symbol.

[7] The 'IsInterface' attribute is set to false for 'PINORDER', 'FLATORDER', 'LEVEL' and 'PARNAME'

SPICE.OUTPUT Property for ViewDraw

Properties ORDER, PINORDER and PREFIX in ViewDraw are translated to the S-Edit SPICE.OUTPUT property. Typical properties and values in ViewDraw are:

```
MODEL = PCH
ORDER = MODEL$ L$ W$ AD= AS= PD= PS=
PINORDER = D G S B
PREFIX = M
```

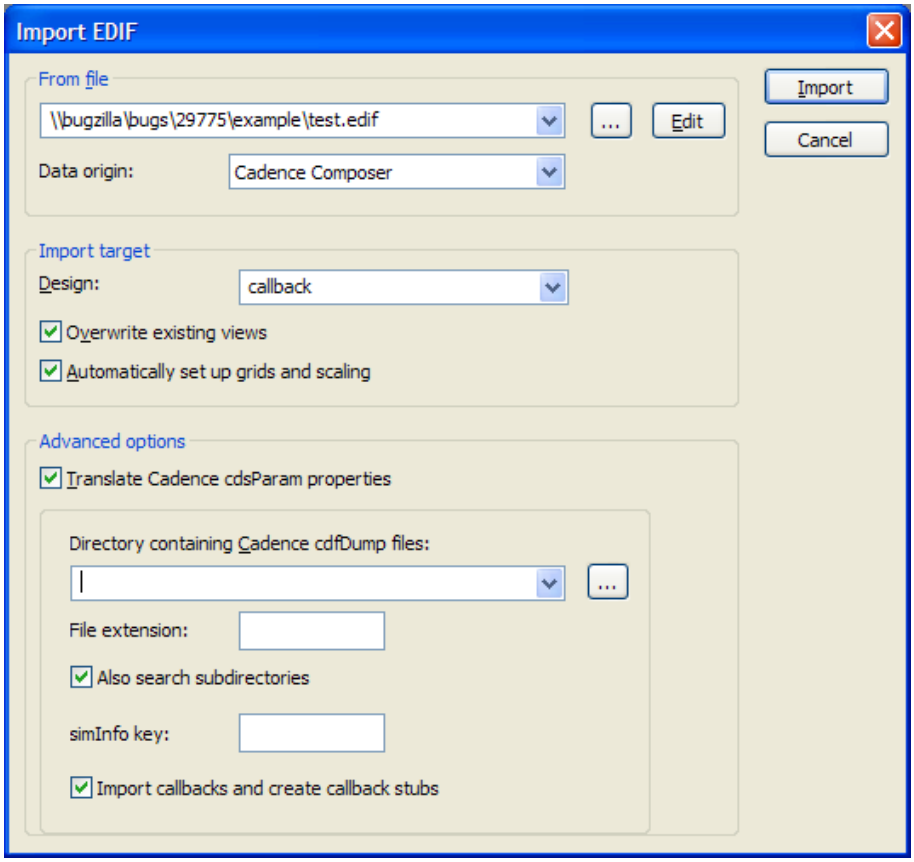
These are translated to the S-Edit SPICE.OUTPUT format, as follows:

```
SPICE.OUTPUT = M${Name} %{D} %{G} %{S} %{B}  $MODEL $L $W [se $AD
   {AD='$AD'}] [se $AS {AS='$AS'}] [se $PD {PD='$PD'}] [se $PS {PS='$PS'}]
```

Tokens in the ViewDraw ORDER property are translated to the SPICE output property as follows:

```
name$ translates to $name
name= translates to [se $name {name='$name'}]
```

Importing EDIF from Cadence Composer



Note that in circuits converted from Composer, the “instNamePrefix” is translated to the “InstanceName” property in S-Edit. (See also “Updating the Name as You Place an Instance” on page 116.)

From file	Specifies the EDIF file to import.
Edit	Opens the specified file in the S-Edit text editor.
Data origin	Select the type of EDIF file from the menu.
Import Target Design:	Specifies the design into which the EDIF file will be imported. If a library exists in the EDIF file with the same name as the design or one of the libraries of the design, then the contents of that EDIF library will be imported into the design or library of the same name. Libraries in the EDIF file that do not match the design name will be created as new libraries of the selected design.
Overwrite existing views	If cells of the same name exist in the EDIF file and the design it is being imported to, a check in this box causes the EDIF file to overwrite the views in the existing cell.
Automatically set up grids and scaling	Causes S-Edit to analyze the contents of the EDIF file for pin spacing and calculate the best grid spacing and scaling.

Translate Cadence cdsParam properties

Cadence EDIF export places the **cdsParam** symbol properties in a separate file from the EDIF files called a **cdfDump** file.

When this box is checked, S-Edit searches the directory specified for **cdfDump** files of the **File Extension** specified, translates the **cdsParam** properties and creates SPICE.OUTPUT when an EDIF file is imported from Cadence. Checking **Also search subdirectories** will do just that for the directory you have entered.

The imported properties will be displayed in the standard S-Edit Properties navigator.

Directory containing Cadence cdfDump files

Specifies the location of the Cadence cdfDump file to be used for translating cdsParam properties.

S-Edit obtains the appropriate library and cell from the header of the file, and applies the property translation to that cell. Parameter files can contain multiple libraries and cells, separated by headers that indicate the library and cell names for the properties that follow. (See "EDIF Translations for Cadence Files," below.)

File extension

When a file extension is entered here, S-Edit reads all files with that extension, regardless of the default base filename.

Also search subdirectories

When this option is checked, S-Edit will search all directories below the one entered.

simInfo key

This option tells S-Edit which simulator in the cdf file to choose for formatting the netlist output, as given by `cdfId->simInfo` in the cdf file, when translating Cadence cdsParam properties.

Import callbacks and create callback stubs

Imports callback function names and parameters attached to schematic views, and generates a list of callback "stubs" with the function name and some basic formatting so that you can enter the function definition. See also **"Callbacks" for Property Values** on page 155.

EDIF Translations for Cadence Files

These translations are performed on all symbols and instances. Electrical ports with names ending with "!" are made global.

cdsTerm(name)

For every cdsTerm label, a property with name "~cds.NNN" is created, with value "[annotate port name]". Here NNN is a unique integer in the symbol scope. The newly created property is placed at the same location as the original label, with the same orientation and justification.

cdsName

For every cdsName label, set the location and text size of the system property "Name" to the values of this label.

cdsParam(n)

If **Translate Cadence cdsParam properties** is checked, *and* a Cadence database directory containing cdfDump files is provided, S-Edit creates properties from the parameter name and value that "**n**" references in the appropriate cdfDump file in the Cadence database.

NLP label Labels containing "**[@**" are converted to properties that reference the corresponding S-Edit properties.

Property values using the Cadence formats pPar, iPar and atPar that refer to other properties are translated to the S-Edit reference format as follows:

pPar("name") ?name

iPar("name") $name

atPar("name") @name

cdsParam File Example

An example of the cdsParam file for a NMOS is shown below.

```
/*************************************************/
 LIBRARY = "tsmc18rf"
 CELL    = "pmos2v"
/*************************************************/
let( ( libId cellId cdfId )
    unless( cellId = ddGetObj( LIBRARY CELL )
        error( "Could not get cell %s." CELL )
    )
    when( cdfId = cdfGetBaseCellCDF( cellId )
        cdfDeleteCDF( cdfId )
    )
    cdfId  = cdfCreateBaseCellCDF( cellId )

;;; Parameters

    cdfCreateParam( cdfId

        ?name           "model"

        ?prompt         "Model name"

    ?defValue      "pch"

    ?type         "string"
    ?editable      "nil"
    ?parseAsCEL     "yes"
  )
  cdfCreateParam( cdfId
    ?name          "macro"
    ?prompt        "Subcircuit name"
    ?defValue      "iPar(\"model\")"
    ?type          "string"
    ?display       "nil"
    ?parseAsCEL     "yes"
```

SPICE.OUTPUT Properties from Cadence Files

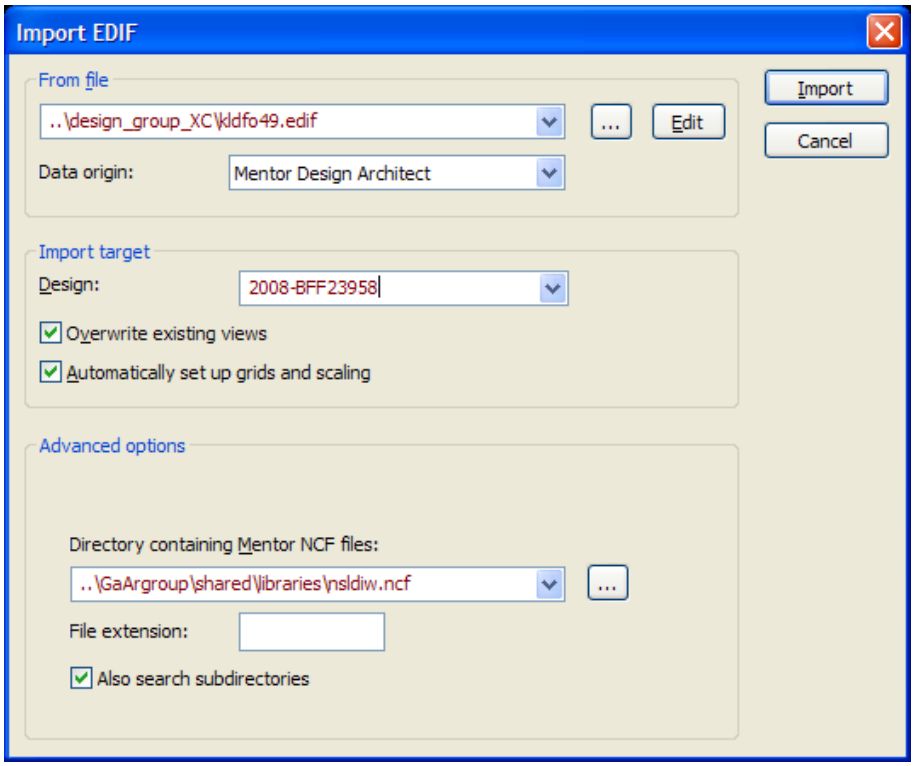
Properties in S-Edit are created from the namePrefix, termOrder, instParameters, and otherParameter properties referenced by Cadence CDF. Given the following properties in the cdfDump file database;

```
namePrefix “M”
termOrder (D G S)
instParameters (m w l ad as pd ps)
otherParameters (bn)
```

the SPICE.OUTPUT property is created as follows:

```
SPICE.OUTPUT = M${INSTANCE} %D %G %S bn m=$m w=$w l=$l ad=$ad as=$as pd=$pd
   ps=$ps
```

Importing EDIF from Mentor Design Architect

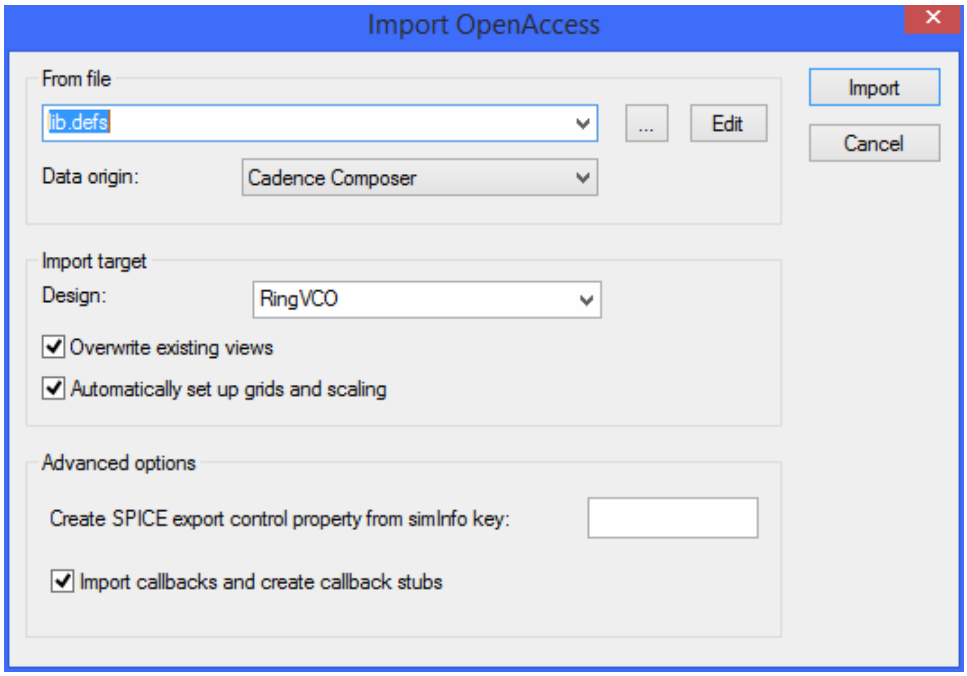


From file	Specifies the EDIF file to import.
Edit	Opens the specified file in the S-Edit text editor.
Data origin	Select the type of EDIF file from the menu.
Import Target Design:	Specifies the design into which the EDIF file will be imported. If a library exists in the EDIF file with the same name as the design or one of the libraries of the design, then the contents of that EDIF library will be imported into the design or library of the same name. Libraries in the EDIF file that do not match the design name will be created as new libraries of the selected design.
Overwrite existing views	If cells of the same name exist in the EDIF file and the design it is being imported to, a check in this box causes the EDIF file to overwrite the views in the existing cell.

Automatically set up grids and scaling	Causes S-Edit to analyze the contents of the EDIF file for pin spacing and calculate the best grid spacing and scaling.
Directory containing Mentor NCF files	Specifies the location of the Mentor NCF file to be used for translation.
File extension	When a file extension is entered here, S-Edit reads all files with that extension, regardless of base filename.
Also search subdirectories	When this option is checked, S-Edit will search all directories below the one entered.

Importing OpenAccess

An openAccess database can be imported into S-Edit. Invoke **File > Import > Import OpenAccess**.



From file	Specifies the libs.def file for the OpenAccess databse to import.
Edit	Opens the specified file in the S-Edit text editor.
Data origin	Select the tool that created the OpenAccess database.
Import Target Design:	Specifies the design into which the OpenAccess database will be imported. If a library exists in the OA database with the same name as the design or one of the libraries of the design, then the contents of that OA library will be imported into the design or library of the same name. Libraries in the OA database that do not match the design name will be created as new libraries of the selected design.

Overwrite existing views — If cells of the same name exist in the OA database and the design it is being imported to, a check in this box causes the contents of the OA database to overwrite the views in the existing cell.

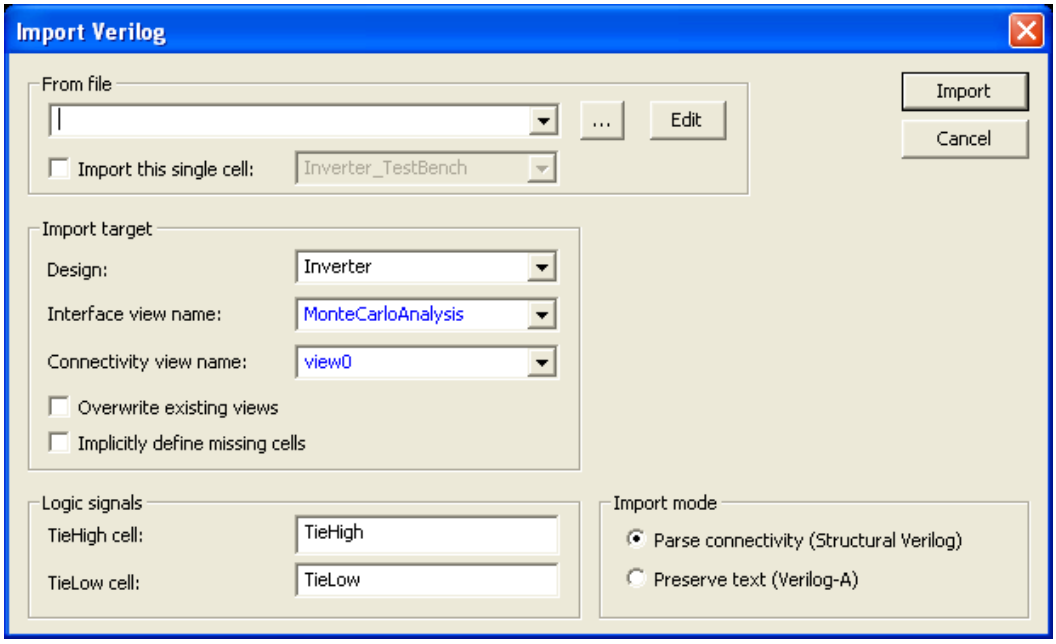
Automatically set up grids and scaling — Causes S-Edit to analyze the contents of the OA database for pin spacing and calculate the best grid spacing and scaling.

Create SPICE export control property from simInfo key — Export control properties are created for all *`cdfId->simInfo`* `simulators listed in the CDF data.` This option tells S-Edit which simulator in the CDF data to convert to the SPICE export control property.

Import callbacks and create callback stubs — Imports callback function names and parameters attached to schematic views, and generates a list of callback "stubs" with the function name and some basic formatting so that you can enter the function definition. Callback "stubs" are only created for callback functions that cannot be found. See also **"Callbacks" for Property Values** on page 155.

Importing Verilog Files

S-Edit will warn when case-insensitive name collisions occur for nets and for instances.



From file — Specifies the Verilog file to import.

Edit — Opens the specified file in the S-Edit text editor.

Import this single cell: — Specify the single cell you want to import.

Note: Bracket symbols **[** and **]** in Verilog instance names will be rewritten as carats **<** and **>** respectively to avoid TCL misinterpretation.

Import Target Design:
Specifies the design into which the Verilog file will be imported. If a library exists in the Verilog file with the same name as the design or one of the libraries of the design, then the contents of that Verilog library will be imported into the design or library of the same name. Libraries in the Verilog file that do not match the design name will be created as new libraries of the selected design.

Import Target: Interface view name
Specifies the interface into which the Verilog file will be imported.

Import Target: Connectivity view name
Specifies the connectivity view into which the Verilog file will be imported.

Overwrite existing views
If cells of the same name exist in the Verilog file and the design it is being imported to, a check in this box causes the Verilog file to overwrite the views in the existing cell.

Implicitly define missing cells
Causes S-Edit to create definitions for cells that are referenced in the Verilog but not present in any library.

TieHigh cell
Specify the name of the TieHigh cell, which will be connected to “1” so that assign statements using constants can be changed to references. Whenever a connection to “0” or “1” is found (either in “assign” statements or in instance pinlists), an instance of the appropriate cell is created, and a connection (possibly via a new net) is generated.

If the TieHigh/TieLow cells do not exist in the library, you will need to define them.

For example:
```
inv x1( .in(0'b0), ...
```
becomes equivalent to:
```
TieLowCell TieLowCell2231( N_2231 );
inv x1( .in( N_2231 ), ...
```

TieLow cell
Specify the name of the TieLow cell, which will be connected to “0”.

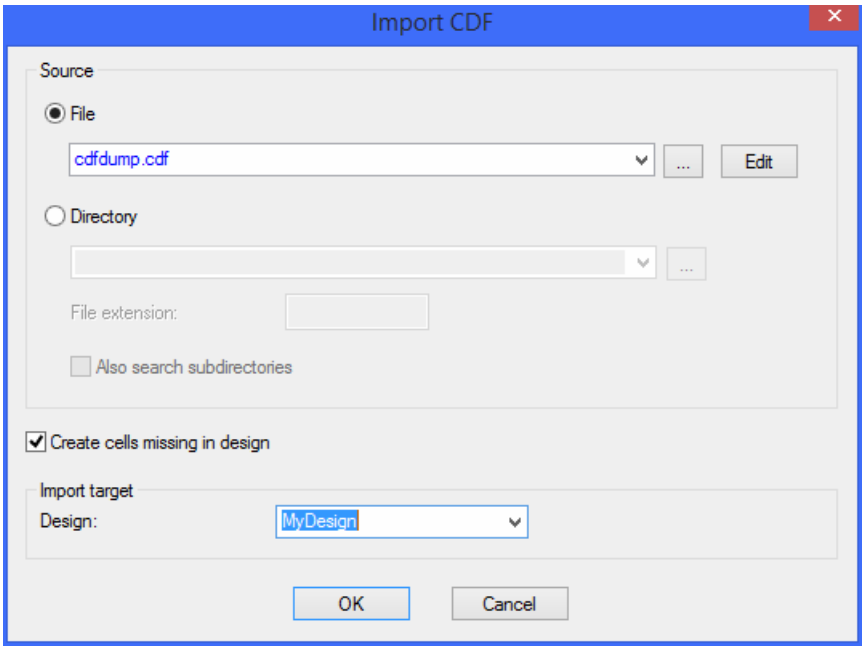
For example:
```
assign out_port = 0'b1;
```
will become equivalent to:
```
TieHighCell TieHighCell332( out_port );
```
where the TieHigh and TieLow cells can be assumed to have a single port, so they can be connected by order and not by name.

Parse connectivity (Structural Verilog)
Parses the connectivity from a Verilog netlist then displays it as Spice views.

Preserve text (Verilog-A)
Copies the text verbatim for each subcircuit from the input Verilog file, as well as the top-level cell, into separate subcircuits. Useful for importing a library of definitions contained in a single file.

Importing CDF Files

Import CDF can be used to import parameters into a Cell from a Cadence Design Systems CDF file.To do this, invoke **File > Import > Import CDF**



File — Name of CDF file to import. A single CDF file may contain parameters for multiple libraries and cells, with headers that indicate the cell and library for the parameters that follow

Directory — Directory name containing CDF files. A directory may contain multiple CDF files, each one contining parameter data for a single or multiple cells.

File Extension — File extension of CDF files to read in the specified directory.

Also search subdirectories — When this option is checked, L-Edit will also read CDF files in subdirectories below the specified directory.

Create cells missing in design — When this option is checked, S-Edit will create a new cell if the cell in the CDF file does not exist in the database.

Import target design — Name of the design into which the CDF file is to be imported.

Exporting a Design

S-Edit can export your design or any portion of it to a variety of netlist formats for simulation or placement and routing. Once you have exported a netlist, you can use T-Spice to simulate it and W-Edit to view the waveform output, directly from S-Edit.

S-Edit writes a netlist for the current cell and all of its instances, unless any of these instances contain output properties that describe them as primitive cells. The following sections provide details on S-Edit's precise export behavior according to the type of netlist being exported.

Before exporting a schematic to EDIF, SPICE, Verilog and VHDL netlists, S-Edit automatically performs a design rule check. If one or more errors are found the export process is aborted.

Exporting SPICE Files

When you export a SPICE netlist file, you can choose several options related to waveform probing. You can also choose whether to suppress the **.END** command in SPICE output. If you plan to include the netlist file in other SPICE netlist files, you should check **Exclude .model**.

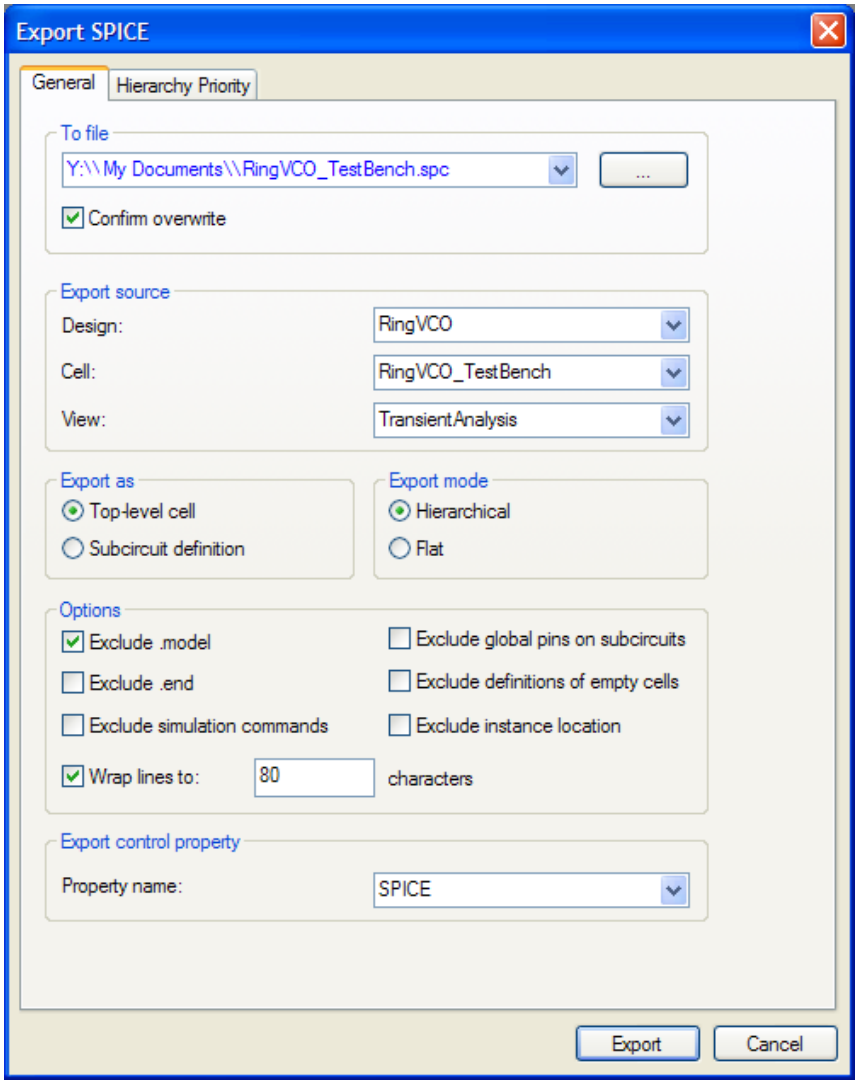
Warning: **Tools > Start Simulation** and **File > Export SPICE**, which both export a SPICE file, are independent and unsynchronized. It is therefore possible to specify two completely different sets of SPICE output properties for the same cell.

Tannerbetical Sort Order

The default sort order is "Tannerbetical," where alphabetic characters are sort first, then any numbers in a name as a number, i.e. C1, C2, C10, C11. Normal alphabetical sort would be C1, C10, C11, C2.

This applies to devices in an exported netlist generated from S-Edit (either through export or using T-Spice); the .op output and small signal parameters in an output file; and within each group of devices with the same SPICE.ORDER value.

To export a SPICE netlist, use the **File > Export > Export SPICE** command.



The simplest way to set a default for the search path, library locations and libraries is to transfer these properties from one cell to another either within or between designs. Simply go to the source cell, and with nothing selected, right-click on the SPICE parameter and chose "Copy tree," then in the cell that you want to transfer the properties to, paste the commands into the command window.

To file

Enter or browse to the filename you wish to export.

The first time a file is exported during an editing session, the default file location is My Documents. After that it will be the one that was used the last time a SPICE file was exported.

Note: A single backslash should not to be used as a path separator in S-Edit, as the evaluator will treat it as an escape character. Path separators should be forward slashes or double backslash.

Confirm overwrite

When checked, the exported file will automatically overwrite an existing file with the same name.

Export Source

Enter the **Design**, **Cell** and **Connectivity View** you want to export.

Export as

- **Top Level Cell**—identify this cell as the top level of your design in the netlist output.
- **Subcircuit Definition**—identify the netlist as a subcircuit using the following format:

```
.SUBCKT name pin1 [pin2 ...] [par1=val1 par2=val2 ...]
<subcircuit definition>
.ENDS [name]
```

Export mode

Choose **hierarchical** or **flat**.

Exclude .model

When checked, excludes the **.model** statement, if any is defined, from the netlist output.

This option only applies to designs with cells having connectivity defined by importing a SPICE netlist. When a netlist is imported, a model definition is created internally for primitive devices. To suppress that model definition from being exported as a .model statement, check this box. It is often desirable to exclude the .model statement in the case where such models are included from external library files. (See also ".OUTPUT property using TCL" on page 184.)

Exclude .end

When checked, suppresses the **.end** statement in the SPICE output file.

Exclude simulation commands

Suppresses S-Edit SPICE simulation setup output from SPICE file export, for example when included files would add transistors or other components that invalidate LVS.

Exclude global pins on subcircuits

When checked, global ports are not expanded in subcircuit definitions and calls and instead a **.global** statement is added to the netlist.

Capped nets are excluded because they have a global definition. Netcaps are ignored. A local net with the same name as a global net will be connected to the global net.

Exclude definitions of empty cells

When checked, excludes definitions of empty subcircuits (which are needed by the SDL router).

Exclude instance location

When checked, does not write the instance location coordinates that are prefaced by a dollar sign to the netlist. (To prevent errors in tools such as Eldo.)

Wrap lines to ___ characters

Enter the number of characters at which output text will wrap to a new line.

Export control property

The property name containing the following sub-properties (see "SPICE Export Properties," below):

- **.OUTPUT**
- **.PRIMITIVE**
- **.ORDER**
- **.DEFINITION**
- **.PREFIX**
- **.PINORDER**
- **.MODEL**
- **.PARAMETERS**

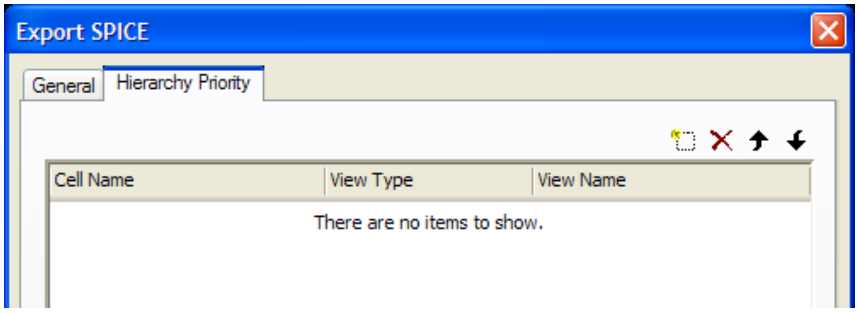
If blank, S-Edit will use the default SPICE.OUTPUT.

Valid separators are space ( ), comma (,), and semicolon (;).

When you export a cell to a SPICE netlist for use with L-Edit DevGen or HiPer DevGen, the Export control property must be set to SDL. This is specific to Tanner EDA libraries; customers should use values based on their libraries. (If "SDL" is not in the drop down list, simply type it in.)

SPICE Export Properties

Note that you can specify the view type that is exported when you export SPICE. If you do not make a choice the default order is schematic, SPICE then Verilog.



SPICE export traverses the design hierarchy from the specified cell, and writes a device instantiation or subcircuit instantiation for every instance in the schematic. For each unique subcircuit instanced, it writes a subcircuit definition. The subcircuit definition consists of a subcircuit header, a device or subcircuit instantiation for every instance in the schematic of that subcircuit, and a **.ends** statement.

SPICE netlisting is controlled by the contents of several properties. Typically, these are placed on a symbol, and provide the default netlisting. They might also be overwritten on a per-instance basis, to allow for custom exports. You can specify multiple SPICE properties and the order of SPICE control properties to use, so that if the specified SPICE.OUTPUT and .ORDER does not exist, the export procedure goes on to the next one in the order. This way, if you want to use a SPICE2.OUTPUT on a symbol, you do not have to have a SPICE2 on all of the symbols.

The properties that control SPICE output are:

- **".OUTPUT property using TCL**" (page 184)**—**format control property for netlist export
- **".PRIMITIVE property**" (page 185)**—**netlist export control property to identify primitive elements
- **".ORDER property**" (page 185)**—**netlist export control property to specify sequential order of exported lines
- **".DEFINITION property**" (page 185)**—**netlist export control property to format subcircuit declarations

These properties must be created as a sub-property of a property. For example, a symbol can have properties where SPICE1.OUTPUT specifies one set of output parameters and SPICE2.OUTPUT can specify a different set of output parameters. Similarly, SPICE1.PRIMITIVE and SPICE2.PRIMITIVE can express different levels at which traversal of the hierarchy should proceed. You would then enter either SPICE1 or SPICE2 in the **Export control property** to specify which .OUTPUT and .PRIMITIVE properties to use when you export SPICE. In most cases there will be only one set of .OUTPUT and .PRIMITIVE properties. (See "Properties Controlling Verilog Export Behavior" on page 194 for further information.)

S-Edit will export a hierarchical SPICE netlist to the specified file according to the following rules:

- If a cell contains a **SPICE.OUTPUT** property on its symbol view, S-Edit will interpret the property value and write its interpreted value to the file, followed by a new line. S-Edit will not search the cell's schematic views for additional instances.

If the cell's symbol view does not contain a **SPICE.OUTPUT** property, S-Edit:

- Writes a subcircuit call with the name of the instanced cell.
- Writes a subcircuit definition by examining the cells instanced on the schematic views in alphabetic order and writing out each instance it encounters.
- Generates the correct **.subckt** and **.ends** lines bracketing each subcircuit. You cannot customize the **.subckt** or **.ends** lines; however, you can change the order in which non-global ports are written.
- If a cell contains instances but no ports or propagated global nets, S-Edit will generate a subcircuit call without ports.
- If a cell contains no instances but contains ports or propagated global nets, S-Edit will write that cell to the netlist as an empty subcircuit.
- If a cell contains no instances, ports, or propagated global nets, S-Edit will ignore the instance.
- S-Edit writes the top-level cell as the main circuit. It is the last block of circuitry S-Edit writes to the file.
- Properties with null values are not exported.

S-Edit will append any properties on a subcircuit symbol except for output properties to the subcircuit definition block as subcircuit parameters.

.OUTPUT property using TCL

The SPICE.OUTPUT property is used to control how instances are written to the SPICE netlist. In particular, the SPICE output property is used to control how terminals, model names, and properties are written in device or subcircuit instantiations. The value of the SPICE output property may contain expressions that reference ports, TCL expressions or other properties. The values of these expressions are then substituted and the results are exported to SPICE. The SPICE.DEFINITION value is always taken from symbol, never from the instance, even if the instance value is more recent.

Evaluation is a two-step process where first properties are substituted then inline TCL code is executed.

A language equivalent to the general property language is used for SPICE output. Braces are optional, and should be used when the property name has a space, or when abutting something immediately after the token. S-Edit searches user properties first, then system properties. The following may be used:

%{N}	Substitute the netname connected to pin N. A reference to a nonexistent pin in the .OUTPUT is referenced to a property at the same level on the instance.
${P}	Substitute the value of property P on the same instance as the SPICE.OUTPUT property.
?{P}	Becomes just P, and references the property P directly above in the hierarchy.
@{P}	Becomes just P, and references a property P above in the hierarchy, searching from the top of the hierarchy down until the value is reached.
%%	A shortcut for all nets connected to the ports of the device in “Tannerbetical” order. (See “Tannerbetical Sort Order” on page 180.)
$$	A shortcut for all the **IsInterface=True** properties of the device.
[string]	Pass the string contained in brackets to the TCl interpreter for evaluation. Use empty brackets [ ] to return an empty string if SPICE output is not desired.

If the SPICE.OUTPUT property is not required and not present, then a SPICE subcircuit call is emitted, and all interface properties are appended. If SPICE.OUTPUT is present, but expands to an empty string, no output is written.

.PRIMITIVE property

A property SPICE.PRIMITIVE property with value TRUE on an instance causes the traversal to stop at this level of the hierarchy (i.e., the schematic page is NOT exported, nor are any subcircuits instanced in it), and causes no definition to be written for the symbol in question.

.ORDER property

A property SPICE.ORDER with real number values on an instance controls the relative order in which SPICE statements are emitted. Instances are written in increasing SPICE.ORDER value. SPICE.ORDER can be a double. See also “Tannerbetical Sort Order” on page 180.

.DEFINITION property

The SPICE.DEFINITION property is used in the definition of a subcircuit. Typically, this is a “.subckt”, with some subset of parameters as shown below.

```
.SUBCKT name pin1 [pin2 ...] [par1=val1 par2=val2 ...]
<subcircuit definition>
```

```
.ENDS [name]
```

The SPICE.DEFINITION value is always taken from symbol, never from an instance. S-Edit will take the symbol property and ignore the overwritten instance property.

Quick SPICE Control Properties

Alternately, you can use the following control properties that do not require the full syntax of the .OUTPUT and .DEFINITION properties.

If there is a conflict, the SPICE.OUTPUT predominates unless a string contains %% or $$. If %% is present, S-Edit looks for a pin order to use if it exists, otherwise all the pins are written out. Similarly, $$ triggers a search for a parameter list; if one is not found all parameters are written out.

.PREFIX property

The SPICE.PREFIX property is used to automatically specify the prefix when writing an instance. For example, the SPICE.PREFIX for a MOSFET would be:

```
SPICE.PREFIX = M
```

.PINORDER property

The SPICE.PINORDER property is used to specify the list of pins written for a device or subcircuit instance, and for a subcircuit definition. The pin list for a MOSFET would be:

```
SPICE.PINORDER = D G S B
```

Buses should be written in the pin list with a specified range in the format <*start*:*stop*> or <*start*:*stop*:*step*>. The order of the bus must be the same as the pin order. For example, a symbol with input bus IN<0:7>, output bus OUT<0:7> and VDD and GND connections:

```
SPICE.PINORDER = IN<0:7> OUT<0:7> VDD GND
```

is equivalent to specifying the following in the SPICE.OUTPUT property:

```
%{IN<0:7>} %{OUT<0:7>} %{VDD} %{GND}
```

.MODEL property

The SPICE.MODEL property is used to specify the model name written for a device or subcircuit instance, or for a subcircuit definition. The value of this property is evaluated and written out.

For example, use the following to write out NMOS as the model name:

```
SPICE.MODEL = NMOS
```

Or, use the following to write the value of the property modelname.

```
SPICE.MODEL = $modelname
```

.PARAMETERS property

The SPICE.PARAMETERS property is used to specify the list of parameters written for a device or subcircuit instance, and also for a subcircuit definition. SPICE.PARAMETERS is a list of parameters, such as:

```
SPICE.PARAMETERS = Prop1= Prop2= Prop3= …
```

This list of parameters will be written as:

```
Prop1=Prop1_value Prop2=Prop2_value Prop3=Prop3_value
```

The following rules apply to SPICE.PARAMETERS:

- A single backslash should not to be used as a path separator in S-Edit, as the evaluator will treat it as an escape character. Path separators should be forward slashes or double backslash.
- An "**=**" after a property name indicates that a property is to be written as PropName=*PropValue*.
- A "**$**" after a property name indicates that only the property value is to be written.
- A "**~**" after a property name indicates that the property is to be written as PropName=*PropValue* if the property value is not the default (symbol) value. If the property value is the default value, then nothing is written.
- The "**=**" or "**$**" characters may be output using "****" as an escape character. "**\=**" outputs "**=**" and "**\$**" outputs "**$**".
- If the value does not exist (empty string) nothing is output for that parameter. (This eliminates the need to write properties with the "**se**" command.)
- If the value is an expression (the operators "**+**", "**-**", "*****" or "**/**") then the value is enclosed in single quotes.

Instance Calls

When writing an instance, If SPICE.OUTPUT does not exist, you can construct one internally as follows and then write out the SPICE.OUTPUT (user-specified or internally constructed) according to the substitutions listed.

```
SPICE.OUTPUT = ${SPICE.PREFIX}${Name} %% ${SPICE.MODEL} $$
```

Where:

[1] If SPICE.PREFIX does not exist, then use '**X**'.

[2] If **%%** exists in the SPICE.OUTPUT then, if a SPICE.PINORDER list also exists, use it to expand **%%**. Otherwise write all pins in Tannerbetical order.

[3] If SPICE.MODELdoes not exist, then use "MasterCell"

[4] If **$$** exists in the SPICE.OUTPUT then, if a SPICE.PARAMETERS list exists, use it to expand **$$**, otherwise write all properties with **IsInterface=True** for **$$**.

For better performance, if the value of the property being written is the default value, then do not write it out.

Subcircuit Definitions

If SPICE.DEFINITION does not exist, construct one internally as shown below, then write out the definition (either the user-specified or an internally constructed one) with the following substitutions:

```
SPICE.DEFINITION = .subckt ${SPICE.MODEL} %% $$
```

Where:

[1] If SPICE.MODEL does not exist, then use “MasterCell”

[2] If **%%** exists in the SPICE.DEFINITION then, if a SPICE.PINORDER list exists, use it to expand **%%**, otherwise write all pins in Tannerbetical order.

[3] If **$$** exists in the SPICE.DEFINITION then, if a SPICE.PARAMETERS list exists, use it to expand **$$**, otherwise write all properties with **IsInterface=True** for **$$**.

SPICE Output Examples

Example 1: MOSFET

A MOSFET symbol will typically have the following properties:

```
SPICE.OUTPUT = M${Name} %D %G %S %B $MODEL W=$W L=$L AS=$AS AD=$AD PS=$PS
    PD=$PD
SPICE.PRIMITIVE = true
```

The symbol will also usually have properties for MODEL, W, L, AS, AD, PS, PD. Consider an instance of a MOSFET with he following properties:

```
MODEL=PMOS
W='24*l'
L='2*l'
AS='114*l*l'
AD='72*l*l'
PS='60*l'
PD='30*l'
```

The name of the instance is P4, and the drain, gate, source and bulk pins of the instance are connected to QB, Q, Vdd, and Vdd respectively.

When the SPICE output statement above is evaluated, the following steps occur:

[1] For $Name, substitute the value of the property “Name” that is on the same instance as the SPICE.OUTPUT property. The property “Name” typically does not exist as a user property, but does exist as a system property, and is the name of the instance, in this case P4.

[2] For %D %G %S %B, substitute the names of the nets connected to ports D, G, S, and B of the instance, in this case QB, Q, Vdd, Vdd.

[3] For $MODEL W=$W L=$L AS=$AS AD=$AD PS=$PS PD=$PD, substitute the values of these properties.

The SPICE device statement written for this instance will then be:

```
MP4 QB Q Vdd Vdd PMOS W='24*l' L='2*l' AS='114*l*l' AD='72*l*l' PS='60*l'
    PD='30*l'
```

The SPICE.PRIMITIVE = true property prevents a subcircuit definition for the MOSFET from being written.

Example 2: MOSFET with Property Substitution

Consider again the SPICE.OUTPUT property for a MOSFET:

```
SPICE.OUTPUT = M${Name} %D %G %S  %B $MODEL W=$W L=$L AS=$AS AD=$AD PS=$PS
    PD=$PD
```

In the above example, the properties MODEL, W, L, AS, AD, PS, PD themselves can use the general property language. For example, the following properties exist on an instance

```
MODEL = PMOS
W = '?Width'
L = '2u'
AS = '$W*3u'
AD = '$W*3u'
PS = '2*$W + 6u'
PD ='2*$W + 6u'
```

And a property Width =20u exists on the instance of the cell in which the MOSFET is located. The SPICE device statement written for this instance will then be:

```
MP_4 QB Q Vdd Vdd PMOS W='20u' L='2u' AS='20u*3u' AD='20u*3u' PS='2*20u +
    6u' PD='2*20u + 6u'
```

Example 3: Conditional Output

In the above example, the properties were assumed to always exist. If a property does not exist, the substitution will result in an empty string. This can produce unwanted results in the SPICE output, such as “AD=” if no value for AD exists.

TCL commands can be inserted into the SPICE output string by placing the TCL command in square brackets. The build in TCL command “se” performs selective evaluation, and can be used to perform conditional output, as in the SPICE.OUTPUT property below:

```
SPICE.OUTPUT = M${Name} %D %G %S %B $MODEL W=$W L=$L [ se $AS {AS=$AS} ] [
    se $AD {AD=$AD} ] [ se $PS {PS=$PS} ] [ se $PD {PD=$PD} ]
```

Here, if a property $AS is resolved with value AS_value, then [se $AS {AS=$AS}] will return AS=AS_value. If $AS does not resolve to anything, because the property AS does not exist, then [se $AS {AS=$AS}] will return an empty string.

Example 4: Controlling Subcircuit Output

The SPICE.OUTPUT property can also be used to control the output of subcircuits, as well as of primitive devices. If no SPICE output statement exists on an instance, then the instance is written as a subcircuit instantiation, with the “X” prefix, and all properties marked as interface properties (sub-property IsInterface=true) are written out.

If needed, the output can be explicitly specified:

```
SPICE.OUTPUT=X${Name} %% ${MasterCell} P1=$P1 P2=$P2
```

Here *Name* and *MasterCell* are system properties referring to the instance name and cell name respectively. %% outputs all the nets connected to the subcircuit ports.

Example 5: Title Block

A title block is an instance that is used to display information about a cell. It references properties and displays information about the cell in which it is instanced. A title block will typically have properties:

```
Cell = ?{Cell}
Info = ?{Info}
Author = ?{Author}
SPICE.OUTPUT=[ ]
SPICE.PRIMITIVE = true
```

The title block uses properties with the ? reference to refer to the parent cell properties. The SPICE.OUTPUT property value [ ] is used to return an empty string, in this case because SPICE output is not desired for this instance.

Passing Subcircuit Parameters to the Originating Cell

When you export a SPICE netlist, S-Edit can pass the values of designated properties in high-level cells down the design hierarchy to the subcircuit definition block of the instanced, or originating, cell.

A property is eligible for subcircuit parameter passing when it meets the following conditions:

- It does not appear on the symbol of a primitive cell—that is, a symbol containing a **SPICE.OUTPUT** property.
- It does not appear on the symbol of the top-level cell.
- Its name is listed in the value of a **SPICE.PARAMETER** property.

Steps in the Subcircuit Parameter Passing Process

[1] Open a high-level cell and switch to a symbol view, if necessary. Add a property whose value contains the parameter you want to pass down the hierarchy.

[2] Add a **SPICE.PARAMETER** property whose value contains the name or names of the properties you wish to pass to the subcircuit.

[3] Switch to a schematic view, select an instance of a cell to which you want to pass the parameter, and replace the value of the originating cell's property with the name of the property in the instancing, or high-level, cell, whose value you want to pass down the hierarchy.

[4] Finally, export a SPICE netlist. S-Edit will include the values of high-level cell properties in the subcircuit definition block of the instanced cell. It will also record property value overrides in *instances.*

Exporting Global Node Connections

When S-Edit exports a SPICE netlist, it writes out global node connections by adding "hidden" ports to each cell's symbol and instances of that symbol. Global connections are thus compatible with any SPICE simulator, without the use of complex node aliasing commands.

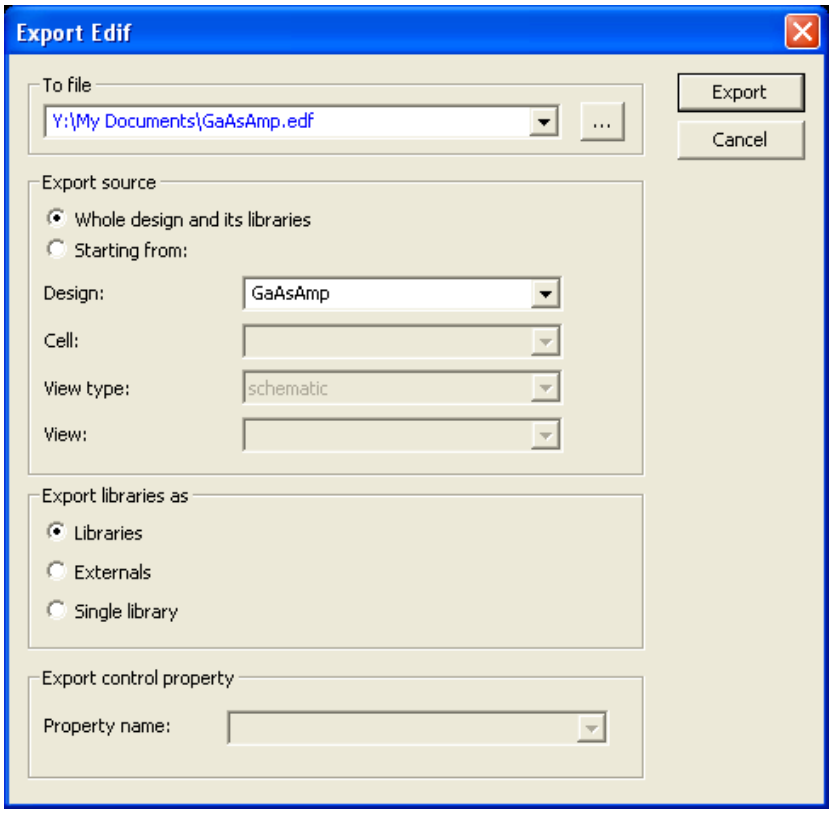
The following is an excerpt from an exported SPICE file containing global nodes. Omitted text is indicated by ellipses (**...**). The subcircuit definition for **OR2** involves user-defined nodes **A**, **B**, and **Y**, and two additional nodes, **Gnd** and **Vdd**, created by S-Edit to propagate the corresponding global nodes into the calls to subcircuit **OR2** inside the definition of **core**.

```
.SUBCKT OR2 A B Gnd Vdd Y
M54 Y N8 Gnd Gnd NMOS W=22u L=2u AS=66p AD=66p PS=24u PD=24u
M55 Y N8 Vdd Vdd PMOS W=22u L=2u AS=66p AD=66p PS=24u PD=24u
M56 N8 B Gnd Gnd NMOS W=22u L=2u AS=66p AD=66p PS=24u PD=24u
M57 N8 A N11 Vdd PMOS W=22u L=2u AS=66p AD=66p PS=24u PD=24u
M58 N8 A Gnd Gnd NMOS W=22u L=2u AS=66p AD=66p PS=24u PD=24u
M59 N11 B Vdd Vdd PMOS W=22u L=2u AS=66p AD=66p PS=24u PD=24u
.ENDS

.SUBCKT core CLOCK DONT_EW ... YELLOW_EW YELLOW_NS
XAND2_1 N4 RED_NS Gnd Vdd GREEN_EW AND2
XOR2_1 TEST_POINT N5 Gnd Vdd N4 OR2
XAND2_2 N4 RED_EW Gnd Vdd GREEN_NS AND2
...
XDFFC1_7 RESETB CLOCK N66 Gnd N65 N66 Vdd DFFC1
.ENDS
```

Exporting EDIF Files

S-Edit can export EDIF schematics retaining the hierarchy, properties and all text. Use the **File > Export > Export EDIF** command. If the top view to export is a SPICE view, export only the netlist. Otherwise, export both schematic and netlist.



To file	Enter or browse for the name of the EDIF file to be output.
Export source	Select the **Whole design and its libraries** you want to export. Or, select **Starting From** and then specify the **Design**, **Cell**, **View type** and **View** of the top level view you want to export.
Export libraries as	**Externals**—outputs references to the interface ports on library elements rather than the entire library. **Libraries**—outputs a list of cells with interfaces. **Single library**—flattens all sub-library cells and references into the design at the top level being exported, with no external libraries or references.
Export control property	Enter the name of the property containing the .PRIMITIVE sub-property used to stop traversal of the design hierarchy.

EDIF export will normally write out an entire design to full extent of the hierarchy. To stop export at a particular instance, place a property with the sub-property .PRIMITIVE having value “true” on the symbol of the cell, or on each instance of that cell.

If a cell's symbol contains an EDIF **.PRIMITIVE = true** property, S-Edit:

- Writes an EDIF cell definition for the cell without examining the cell's schematic pages for additional instances. The cell definition will contain ports and global ports.
- Treats the cell as if it were instanced in the top-level cell.

If a cell does not contain an EDIF **PRIMITIVE = true** property on its symbol page, S-Edit examines all of the cell's schematic and writes out each instance to the netlist.

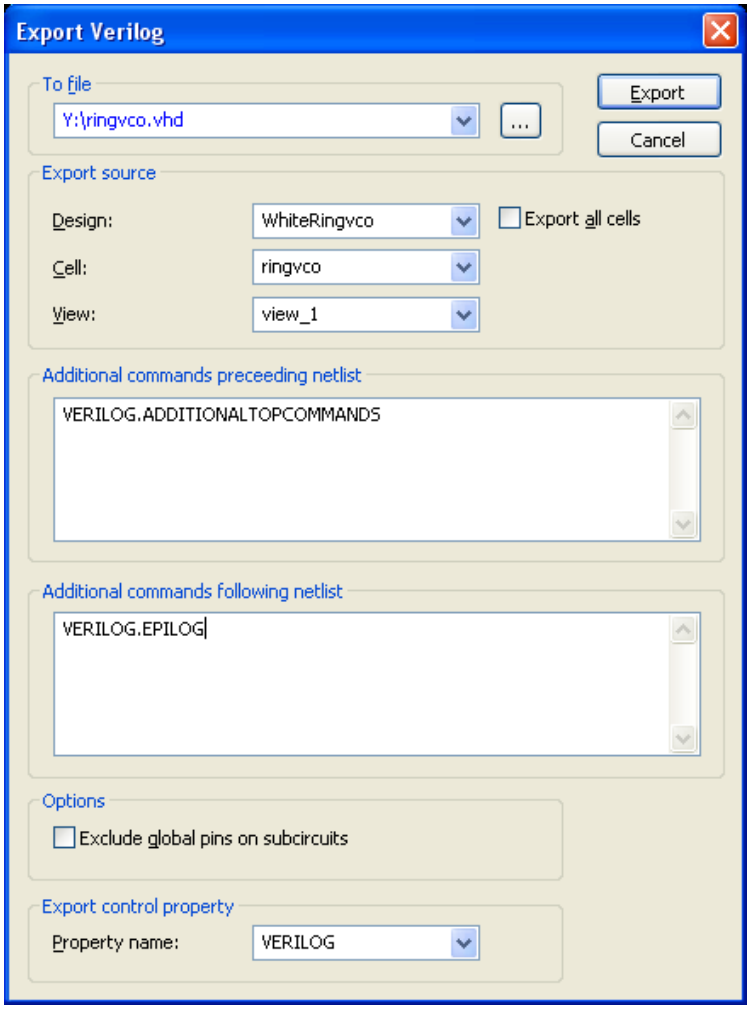
- If the instanced cell contains no ports or propagated global nets, S-Edit ignores the instance.
- If the instanced cell contains ports but no instances, it is an error. S-Edit writes a message to the netlist file identifying the cell with the error and stating that it “requires an EDIF definition.” S-Edit also displays a dialog box to warn you of the error.
- After writing all primitive cell definitions, S-Edit writes a top-level cell containing instances of all primitives in the design and the nets that connect these primitives.
- S-Edit generates scoped node names, which uniquely identify a node by including its hierarchical position in the node name-e.g., *inst1/inst2/.../node_name*, where *inst1*, *inst2*, etc. are the names of the instances in descending order from the top level of the design to the level of the node, and *node_name* is the name of the node. Local node names are scoped to show the hierarchy of instances from the top level to the level that contains the node. Global node names are scoped to show the hierarchy of instances from the cell containing the global node symbol to the level at which the global node is capped. Uncapped global nodes will be written without a scope-that is, **Gnd** will simply appear as “Gnd” in the output file.
- S-Edit will automatically convert any names that are incompatible with EDIF naming requirements to a legal EDIF name using the **rename** construct.

Example

A netlist-only EDIF file that terminates at standard cells, rather than going to transistor level, is desirable when you export EDIF for standard cell place and route. To do this, create a property EDIF.PRIMITIVE = true (the period indicates PRIMITIVE is a sub-property of EDIF) on each standard cell symbol, and then enter EDIF as the **Export control property**.

Usually an EDIF schematic that traverses all the way down the hierarchy is the desired output when you export EDIF to view in another tool. In this case, leave the **Export control** property blank.

Exporting Verilog Files



To file	Enter or browse for the name of the netlist file to be output. Valid file types are **.v**, **.va** and **.verilog**.
Export source	Enter the **Design**, **Cell** and **Connectivity View** you want to export.
Export all cells	Check this box to export all cells in a design.
Additional commands preceding netlist	Properties in this field are written verbatim after the title block, on a single line.

Additional commands following netlist	The values in this field are written at the end of the Verilog output.
Options	Use the **Export global pins on subcircuits** checkbox to ignore global ports in the netlist during export so they are not exported as parameters.
Export control property	Enter the **Property name** containing the .PRIMITIVE sub-property used to stop traversal of the design hierarchy.

Properties Controlling Verilog Export Behavior

In some cases, you might want to identify an element which contains a schematic view as a primitive element. This situation most typically arises with standard-cell libraries that contain cells which have transistor-level schematics attached.

To identify a cell that contains a schematic as a primitive, such as a NAND cell, place a property on the symbol of the cell with a subproperty named PRIMITIVE and value TRUE, and identify that property name as the export control property in the **Export Verilog** dialog. Typically you would create a property called VERILOG, with subproperty PRIMITIVE and value TRUE, but any name can be used in place of “VERILOG.”

VERILOG.OUTPUT

The Verilog exporter is controlled by several user properties including VERILOG.OUTPUT, VERILOG.PRIMITIVE and VERILOG.DEFINITION. The "VERILOG" part of these properties may be substituted by a user-supplied prefix in the **File > Export > Verilog** dialog. Note that VHDL and SPICE operate the same way.

The Verilog output option in S-Edit creates a subcircuit call for every level in the design hierarchy. S-Edit cells that have symbols but no schematics are considered primitive elements, and no definition of these elements is written to the Verilog output file. It is assumed that these primitive elements are contained in user-supplied Verilog libraries, if they are required at all.

If an instance contains a VERILOG.OUTPUT property, the property will be evaluated and exported to the Verilog output file, instead of the default subcircuit call. In the special case of an empty string (obtained by setting the property value to [], which evaluates in TCL to “nothing,”) no carriage return should be appended therefore it's as if the item doesn't exist at all. Like all user properties, VERILOG.OUTPUT is inherited from the symbol definition.

VERILOG.OUTPUT is primarily useful for exporting comments, global statements, and special-purpose output statements. **VERILOG.OUTPUT = []** is also used with **VERILOG.PRIMITIVE = true** to prevent a symbol or cell (for example a title block, SPICE control instance, or cell like VDD) from being exported.

The Verilog exporter recognizes two properties on a schematic view and outputs their values when exporting Verilog. The value of VERILOG.PROLOG, similar to Spice.General.AdditionalCommands, is written verbatim after the title block, on a single line. For example, you can use it to instruct the Verilog netlister to add a timescale statement when netlisting. VERILOG.EPILOG outputs the value at the end of the export.

VERILOG.PRIMITIVE

If an instance contains a VERILOG.PRIMITIVE property, the instance is written out as usual, but no definition for the cell is written. Note that it is the presence of the VERILOG.PRIMITIVE property, and not its value, that causes this behavior.

To prevent a symbol or cell (for example a title block or VDD or GND cell) from being exported, use **VERILOG.PRIMITIVE = true** so a definition is not generated, and **VERILOG.OUTPUT = []** so a subcircuit call is not generated.

VERILOG.DEFINITION

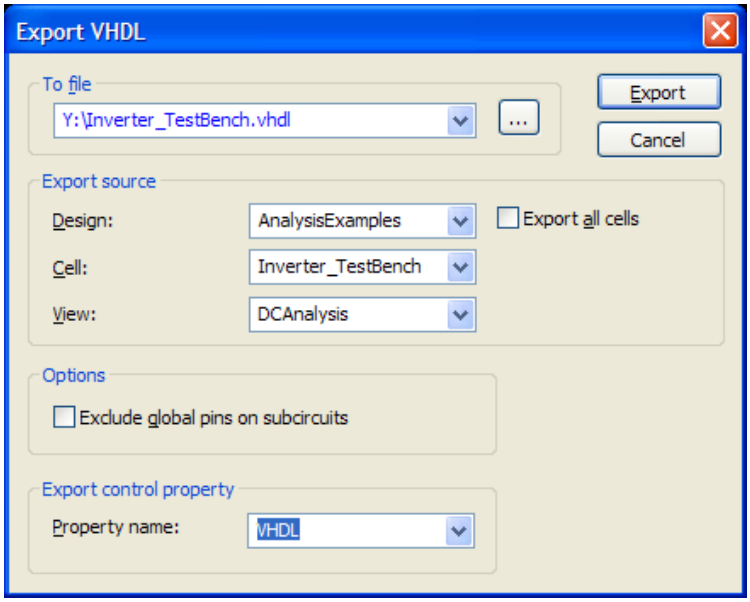
If a symbol contains a VERILOG.DEFINITION property, that property is evaluated and exported, one time, as a definition. This is primarily useful for conditional inclusion of files and/or external models.

VERILOG.MODEL

Lastly, we have a VERILOG.MODEL property that is evaluated and used to replace the cellname in both the definition and each instance, unless VERILOG.PRIMITIVE, .DEFINITION, or .OUTPUT properties exist—if they do, they have precedence.

Exporting VHDL Files

Note that by default, VHDL port mapping violations are set to **Ignore**



To file — Enter or browse for the name of the netlist file to be output.

Export source — Enter the **Design**, **Cell** and **Connectivity View** you want to export. Check the **Export all cells box** to export all cells in a design.

Options — Check the box to **Exclude global pins on subcircuits** from the export.

Export control property Enter the **Property name** containing the .PRIMITIVE sub-property used to stop traversal of the design hierarchy.

Properties Controlling VHDL Export Behavior

S-Edit will write the connectivity information in a design in VHDL format according to the following rules (see also "Properties Controlling Verilog Export Behavior" on page 194):

- If an instanced cell contains a VHDL.PRIMITIVE property on its symbol view, S-Edit will write a VHDL entity with an empty behavioral architecture, which you can edit in the VHDL code to provide a behavioral definition. S-Edit will not search the cell's schematic views for additional instances.
- To prevent a symbol or cell, for example a title block or a SPICE control instance, or cells like VDD, from being exported, use **VHDL.PRIMITIVE = true** so a definition is not generated, and **VHDL.OUTPUT = []** so a subcircuit call is not generated.
-

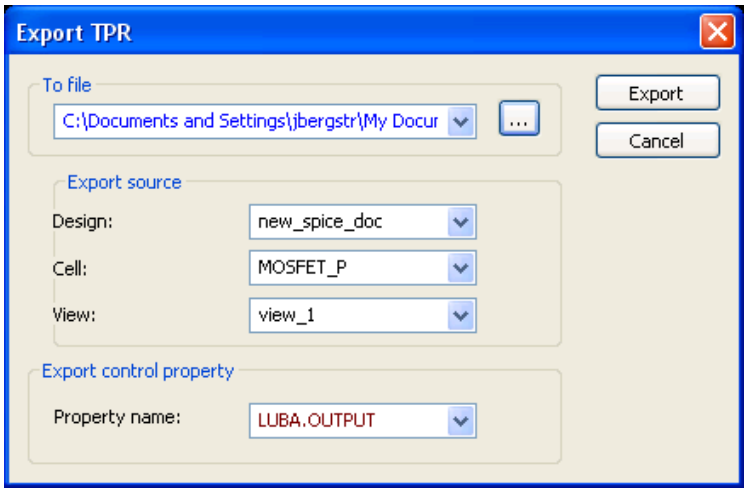
If the instanced cell's symbol view does *not* contain a VHDL.PRIMITIVE property, S-Edit:

- Writes a structural architecture with the name of the instanced cell.
- Writes a structural architecture by examining the cells instanced on the schematic views and writing out each instance it encounters, plus the connectivity of the instance, so that you can conveniently provide a description in VHDL.
- Defines the primitive as a VHDL entity with ports defined as std_logic signal ports.
- If an instanced cell contains no instances, S-Edit will ignore the instance
- If an instanced cell contains ports but no instances, S-Edit will signal an error and write the cell to the netlist as an entity with empty structural architecture.
- S-Edit writes the top-level cell as the top-level entity in the VHDL file.

Note: Unlike other netlist formats, VHDL netlists require that the top-level cell have a valid symbol view, with ports that correspond to its schematic ports. This information is used to define the top-level entity.

Exporting TPR Files

TPR is a flat netlist format you can use to place and route your design in L-Edit.



To file Enter or browse for the name of the netlist file to be output.

Export source Enter the **Design**, **Cell** and **View** you want to export.

Export control property Enter the name of the output property, typically simply TPR.

If an instanced cell contains a TPR.OUTPUT property, S-Edit will generate a netlist according to the following rules. It will not examine the schematic for additional instances.

- The .OUTPUT property can take two values, “C” and “CP,” where “CP” is used exclusively for pad cells. These must have a terminal called “Pad.”
- S-Edit generates two lines for the cell and writes them to the file. The first line indicates formal parameters for the instanced cell, and the second line indicates the mapping of formal parameters to node names in the instance.
- S-Edit will not generate global node ports for the cell.
- The property TPR.OUTPUT can have any prefix, which is then entered in the **Export control property** field.
- S-Edit generates scoped names which uniquely identify an instance by including its hierarchical position. (For example, top/cell1/cell2/.../instance_name, where cell1, cell2, etc. are the names of the cells in descending order from the top level of the design to the level containing the instance.) Global symbol names are scoped to show the hierarchy of instances from the cell containing the global symbol to the level at which the global net is capped. Uncapped global nets will be written without a scope—that is, Gnd will simply appear as “Gnd” in the output file.
- Instances with TPR.OUTPUT = <*empty value*> are not exported.
- If an instanced cell does not contain a TPR.OUTPUT property on its symbol view, S-Edit continues to examine all cells in alphabetic order and writes out each instance to the netlist.
- If the instanced cell contains no ports or propagated global nets, S-Edit ignores the instance.
- If the instanced cell contains no instances but does contain ports, it is an error. S-Edit writes a message to the netlist file identifying the cell with the error and stating that it “requires a TPR definition.” S-Edit also displays a dialog box warning of the error.

TPR File Format

TPR files are ASCII text files that are generated automatically by S-Edit; they can also be created with any text editor.

Syntax

A portion of the **.tpr** netlist file for the bargraph example is shown below.

Comment line

```
$ TPR written by the Tanner Research schematic editor, S-Edit
$ Version: 2.0 Beta 5     Jan 7, 1998  16:07:16
```

Pad cell definition
Instance definition

```
CP PadOut DataOut Pad;
UPadOut_1 N2 PAD_B1_L31;

:

CP PadInC DataIn DataInB DataInUnBuf Pad;
UPadInC_1 N68 IPAD_9/N2 IPAD_9/N1 PAD_L9_SCO;
```

In the two lines above, **DataIn**, **DataInB**, and **DataInUnBuf** are the names of ports in the pad cell **PadInC** (PortList). **N68**, **IPAD_9/N2**, and **IPAD_9/N1** are the names of nets attached to these ports (NetList). **PAD_L9_SCO** is the name given to the body region of the pad. **L9** identifies the position of the pad as the ninth pad from the top on the left side of the padframe.

```
:
```

Ground pad

```
CP PadGnd Pad;
UPadGnd_1 PAD_R8_GND;
```

Power pad

```
CP PadVdd Pad;
UPadVdd_1 PAD_L6_VDD;

:
```

Cell definition
Instance definition

```
C INV A Out;
UINV_3 BARGRAPH_1/BG64_2/N9 BARGRAPH_1/BG64_2/SFT3;

:

C Mux2 A B Out Sel;
UMux2_1 BARGRAPH_1/BG64_1/BG4_1/N118 BARGRAPH_1/BG64_1/BG4_1/N108
N62
+  BARGRAPH_1/BG64_1/S11;
```

In the three lines above, **A**, **B**, **Out**, and **Sel** are ports in the standard cell **Mux2** (PortList). **BARGRAPH_1/BG64_1/BG4_1/N118**, **BARGRAPH_1/BG64_1/BG4_1/N108**, **N62**, and **BARGRAPH_1/BG64_1/S11** are the names of nets attached to these ports (NetList). Note that these net names include the hierarchical structure of the schematic. This is the manner in which S-Edit creates a "flattened" **.tpr** netlist. A plus sign (**+**) indicates a continuation of the previous line.

Interpretation

Pad cells are defined in the format:

```
CP <padname> <pin1> <pin2> … Pad
U<gateUID> <net1> <net2> … Pad_<PadPosition>
```

Standard cells are defined in the format:

```
C <cellname> <pin1> <pin2> …
U<gateUID> <net1> <net2> …
```

A **.tpr** file must conform to the following rules:

- All signals which are to be routed within the core or from the core to the padframe are required to be listed, with the exception of the Vdd and Gnd signal connections to pads.
- For each cell, the **PortList** and **NetList** must have the same number of elements.
- The name “PAD” in the PortList of a pad cell refers to the actual bonding region of the pad, and is not actually involved in the placement and routing process. Pad cells must have a signal marked “PAD.”
- The bonding region of a pad can contain the location of the pad on the padframe. For example, “B1” stands for the left-most pad on the bottom side of the padframe (L = Left, B = Bottom, R = Right, T = Top).
- Power and ground pads do not have to be included in the netlist. If they are not included, SPR will place them automatically.

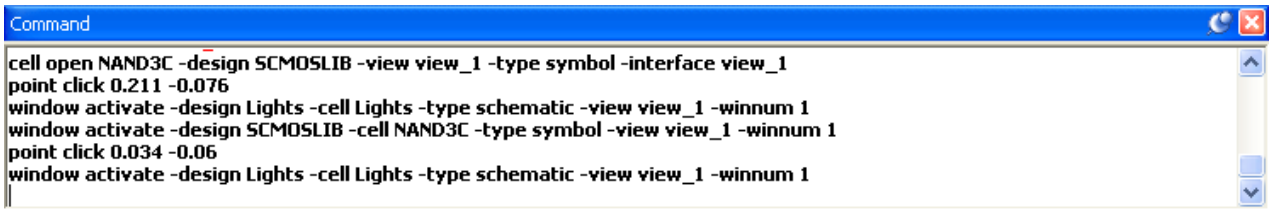
The parts listed in the file must match the cells contained in the layout library. To match, the name of the part must be identical to the name of the library cell (except for case), and every signal listed in the part description must have at least one port of the same name somewhere in the library cell.

9 Scripting with TCL

S-Edit uses the TCL scripting language to execute all operations. When you initiate an operation in the S-Edit graphical interface, the corresponding TCL command is written to the Command window.

In fact, it is the TCL commands in the Command window that cause S-Edit to execute an operation. Therefore, you can also write scripts directly into the Command window to execute operations.

Text can be written in directly, copied from executed operations and then pasted back into the Command window, or invoked from a saved TCL file. Or you can drag and drop a TCL file from a browser into the Command window to execute it immediately.


Command
cell open NAND3C -design SCMOSLIB -view view_1 -type symbol -interface view_1
point click 0.211 -0.076
window activate -design Lights -cell Lights -type schematic -view view_1 -winnum 1
window activate -design SCMOSLIB -cell NAND3C -type symbol -view view_1 -winnum 1
point click 0.034 -0.06
window activate -design Lights -cell Lights -type schematic -view view_1 -winnum 1

You can also run scripts automatically when a design is loaded and when S-Edit opens or closes by placing TCL files in certain special directories (see “Executing Scripts Automatically” on page 203).

S-Edit TCL Command Help System

S-Edit provides a help system for its available TCL commands, which is displayed in the Command window as follows.

help-help	Displays help information for the “help” command itself.
help	Displays a list of all TCL commands available in S-Edit.
help all	Displays all help information for all commands.
***<command>* -help**	Displays the options and syntax for a specific command.
help -search *<list of keywords>*	Displays command header and matching options (if any). For example, “help -search miter” would print the following (and perhaps other lines, if “miter” is an option to other commands): # path -- path command # -join miter | round | bevel | layout: join style # -miter <miter angle>: angle in 1e-6 degrees

TCL Commands Available in S-Edit

The available S-Edit commands are listed below.

allowselect	Control objects that can be selected
cell	Cell command
copy	Copy view selection to clipboard
cut	Cut view selection to clipboard
database	Database command
delete	Delete command
design	Design command
document	Document command
draw	Draw command
dtos	Double to string. Used with the “-decimal N” optional parameter to control the number of digits after the decimal point. If N begins with a zero, it adds zeros to the end of the result to always display N number of digits after the decimal. For example, given 50.404177, then [dtos 50.404177 -decimal 2] = 50.4 while [dtos 50.404177 -decimal 02] = 50.40. Similarly, the argument “-pad” causes optional spaces to be added to the left-hand side of the output, such that decimal points align. dtos supports an argument “-format” which can take on values "spice" (the default, which uses “X”), "hspice" (which uses “MEG”), "si" (which uses “M”), and "abridged_si" (which uses “M”, and differs from si by not going to some of the more exotic scales such as yotta). This argument can be used in conjunction with the “-saveformat” option to set the default formatting for "dtos". Note that these settings do not affect any other property evaluation code path. In other words, the **Export > Spice netlist** is still written with "X", as is the evaluated property mode.
duplicate	Duplicate selected data in a view
exit	Exit command
export	Export command
find	Finds objects, nets, etc.
help	Displays a list of the TCL commands related to S-Edit
import	Import command
instance	Instance command
mgc_rve_export_netlist	Export netlist
mgc_rve_probe_sinst:	Select instance to probe.
mgc_rve_probe_snet	Select net to probe.

mode	Mode command
netcap	Netcap command
paste	Paste clipboard contents to a view
path	Path command
point	Point command
port	Port command
print	Print command
probe	Probe net or terminal
property	Property command
redo	Redo command
report	Report command
setup	Setup command
stod	String to double
test	Test command
texteditor	Texteditor command
textlabel	Textlabel command
tsource	Source override
undo	Undo command
visible	Control object visibility
window	Window command
workspace	Workspace command

Running TCL Scripts

To run a TCL script in S-Edit, you can:

[1] Drag and drop the script into the Command window from a browser.

[2] Invoke **File > Open > Execute Script**

[3] Invoke **File > Recent Scripts** and select a previously run script.

[4] Type "**source**" followed by the path and filename for the script, as described in "Source Scripts," below.

(To open a TCL file in the S-Edit text editor without executing it, or to open any other kind of text file, use **File > Open > Open File**.)

Source Scripts

Source scripts take the form:

```
source filename | foldername [-subfolders] [-relativeto user|design]
   [-mru false|true]
```

The source command logs a warning when S-Edit cannot locate a specified file. If you are not seeing these warnings it could be that your verbosity setting is below that of the warning. To increase the verbosity setting, right-click on the log window and choose Customize; the verbosity setting is in the Filters tab.

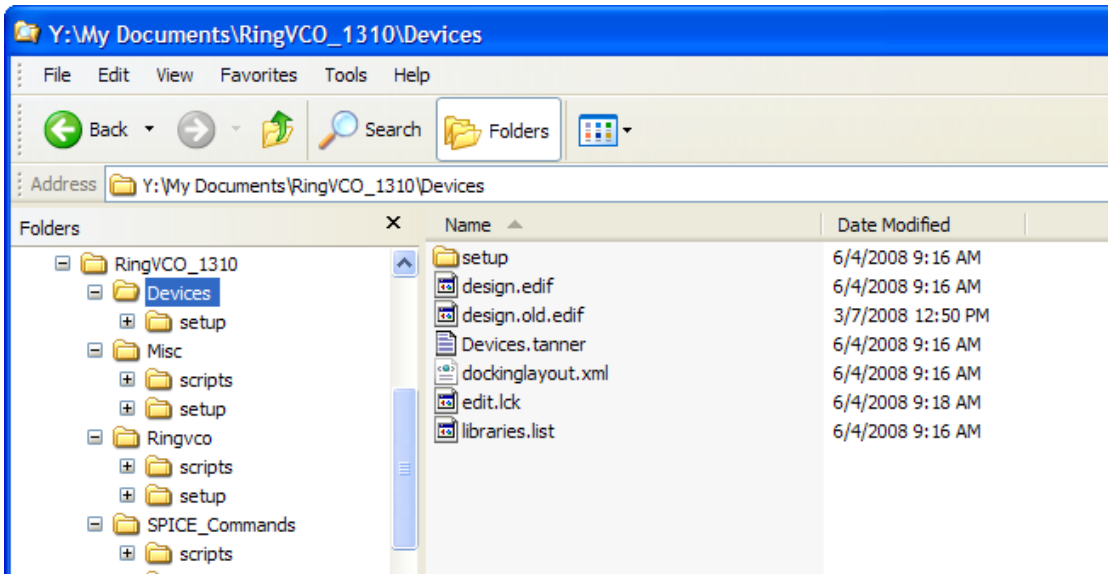
filename	Source runs the specified script file. The filename can contain an absolute or relative path.
foldername	Source runs all scripts in the specified folder in alphabetical order. The foldername can contain an absolute or relative path. Does not recurse into subfolders by default.
-relativeto user | design	**-relativeto user** If a relative path is given, and **-relativeto user** is specified, the root for a windows installation is **C:/Documents and Settings/<*username*>/Application Data/Tanner EDA/scripts**. **-relativeto design** If a relative path is given, and **-relativeto design** is specified, the root is the design folder. Note that “source -path test -relativeto design” runs all the scripts in the folder “<*design*>/scripts/test” folder. (The “scripts” folder name is appended to the <*design*> folder name.) **-relativeto *user*** is the default if no **-relativeto** option is specified.
-subfolders	When a subfolder is specified, the source search path recurses breadth first into subfolders. (All scripts in the specified folder are run first, in alphabetical order, then subfolders are recurred into in alphabetical order and scripts within each subfolder are read alphabetically.)
-mru false | true	By default, “source filename” is set to **-mru true**, so that script is placed in the list of most recently used (**mru**) files. The default setting for “source foldername” is **-mru false**, it does not put the scripts in that folder into the mru. Specifying “-mru false” or “-mru true” allows you to override the default behaviors.
-help	Displays an explanation of command options and syntax.

Executing Scripts Automatically

Each library in S-Edit has several folders that it reads automatically.

Though file naming is unrestricted, S-Edit reads files in alphabetical order. Thus if you have a required sequence, use file names to control the order in which scripts in any folder are executed.

Library references in the **libraries.list** file that use absolute path names are saved and loaded with a design.



Running a Script when S-Edit Launches — **startup**

To run a script automatically when S-Edit opens, save the script to the following folder. Scripts saved in the "**startup**" directory will run on any design that is opened.

.../<*username*>/Application Data/Tanner EDA/scripts/startup

For Vista operating systems use:

`C:/Users/<username>/AppData/Roaming/Tanner EDA/scripts/startup`

Running a Script when S-Edit Closes — **shutdown**

To run a script automatically when S-Edit closes, place the script in the folder. Scripts saved in the "**shutdown**" directory will run on any design that is opened.

.../Documents and Settings/<*username*>/Application Data/Tanner EDA/scripts/shutdown

For Vista operating systems use:

`C:/Users/<username>/AppData/Roaming/Tanner EDA/scripts/shutdown`

Running a Script when a Design Opens — **open.design**

To run a script when any design is opened, place it in the following folder.

C:/Documents and Settings/<*username*>/Application Data/Tanner EDA/scripts/open.design

S-Edit also creates an **open.design** folder under each design folder. All scripts placed in this directory will execute automatically whenever that specific design opens.

.../<*designname*>/scripts/open.design

Load Order for TCL Files

When you open a design, S-Edit reads TCL files and folders in the order described below. At its simplest, this sequential load order allows for a universal setup intended for all users of a given design and also for individual users to modify the universal setup by saving their own setup preferences to a user preferences folder.

Scripts in any folder are executed in alphabetical order of file name. Therefore, if you have a required sequence, use file names to control the order in which your scripts are executed.

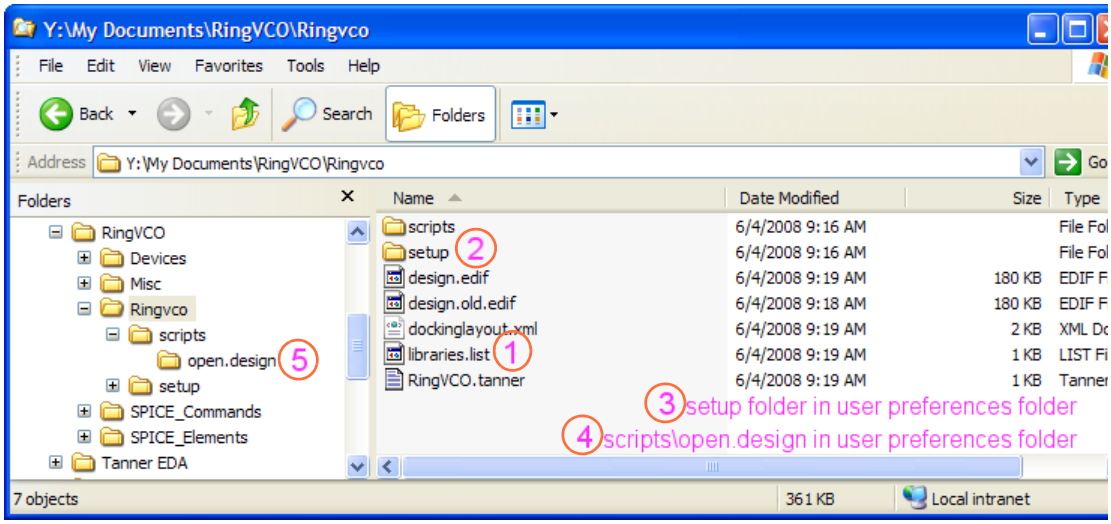
[1] Libraries in the design folder [.../*designname*/**libraries.list**] file are loaded first, using *depth-first recursion* (see "Load Order for Libraries," below.)

[2] Setup scripts are read from the **design folder** [.../*designname*/**setup**]

[3] Setup scripts are read from the **user preferences folder** [**C:/Documents and Settings**/<*username*>/**Application Data/Tanner EDA/scripts/open.design/setup**]

[4] If one is defined, a source statement in [.../*username*/**scripts/open.design**] runs scripts in that folder relative to *user* settings.

[5] If one is defined, a source statement in [.../*designname*/**scripts/open.design**] runs scripts in that folder relative to *design* settings.



Load Order for Libraries

When a library is opened because it is listed in the **libraries.list** file as in step 1 above, S-Edit uses depth-first recursion to read TCL files as follows:

[1] Setup scripts in the library's design folder [.../*libraryname*/**setup**]

[2] Setup scripts in the user preferences folder [.../*username*/**scripts**/**open.design**/setup]

[3] All other scripts in the user preferences folder [.../*username*/**scripts**/**open.design**]

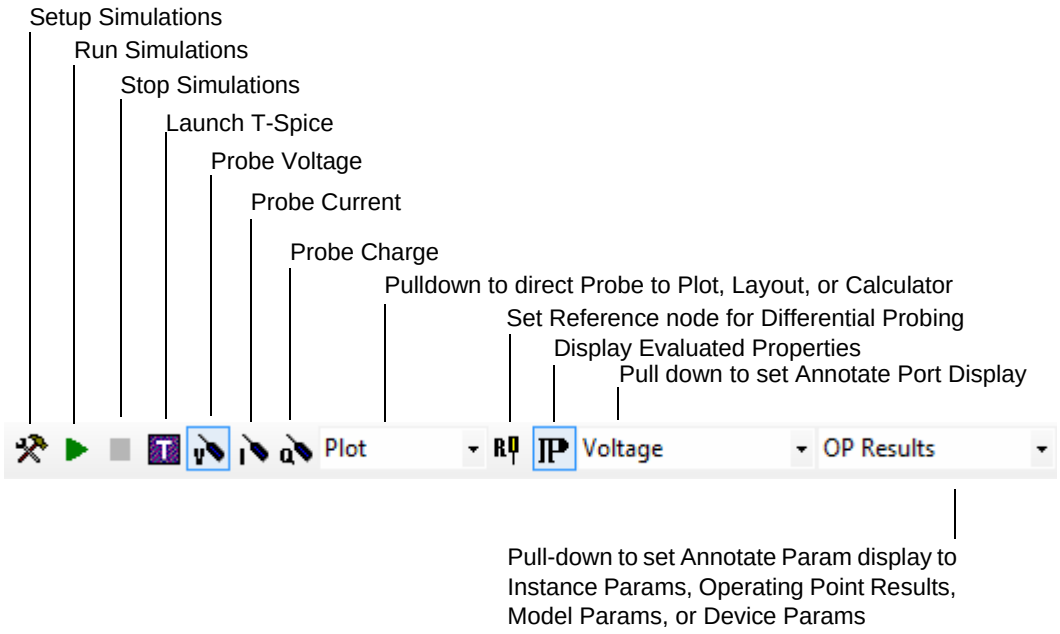
[4] All other scripts in [.../*libraryname*/**scripts**/**open.design**]

10 Simulation and Waveform Probing

Tanner's schematic and simulation tools are fully integrated to allow AC, DC, or transient analysis of your design, with interactive setup, simulation, and post-processing. The three components of this process are S-Edit, T-Spice, and W-Edit, and there are three primary stages to the simulation flow:

- In the *setup* stage, you enter commands and information which describe the type of simulation (DC, AC, transient, etc.), and establish the simulator options and outputs.
- In the *design export* and *simulation* stage, S-Edit generates a SPICE file (a netlist) from the design. Then, T-Spice simulates the SPICE file to create a probing data file with voltage and current values for each node and device in the design, and for each analysis specified in the SPICE file.
- In the *probing* stage, W-Edit displays traces from the probe data file corresponding to an analysis type and a specific net or device selected in S-Edit. S-Edit can also annotate your schematic with node voltages and device terminal currents and charges.

The SPICE simulation toolbar provides quick access to key functions.


Setup Simulations
Run Simulations
Stop Simulations
Launch T-Spice
Probe Voltage
Probe Current
Probe Charge
Pulldown to direct Probe to Plot, Layout, or Calculator
Set Reference node for Differential Probing
Display Evaluated Properties
Pull down to set Annotate Port Display
Plot
Voltage
OP Results
Pull-down to set Annotate Param display to Instance Params, Operating Point Results, Model Params, or Device Params

SPICE Simulation Settings

Before running the T-Spice simulator you must enter some basic settings which will define the type of simulation to be performed and the outputs from the simulation. You enter these values in the **Setup SPICE Simulation** dialog.

Note that simulations are performed on the active cell, as opposed to an entire design.

All Fields in Setup SPICE Simulation are Evaluated

All fields in the **Setup SPICE Simulation** dialog are evaluated. Evaluating fields in the **Setup SPICE Simulation** allows for customizing, such as setting the output file name based on the name of the cell being simulated.

Because the back slash character (****) is an escape-sequence prefix in S-Edit, path names in this dialog should use forward slashes.

S-Edit converts back slash (****) and double back slash (****) characters to a single forward slash (**/**) in path names when you use the browse button. Similarly, S-Edit converts backslashes to forward slashes in path names when you import files created with older versions of S-Edit.

Characters followed by a backslash are evaluated simply as the character, as long as the character has no special escape sequence meaning. For example, evaluation of "\w" is "w". In order to have a backslash (****) in the output, for example as a file name delimiter, you can use double backslashes (e.g. c:\\dave\dave.sp) or you can use forward slashes instead, (e.g. c:/dave/dave.sp.)

Importing and Exporting SPICE Simulation Settings

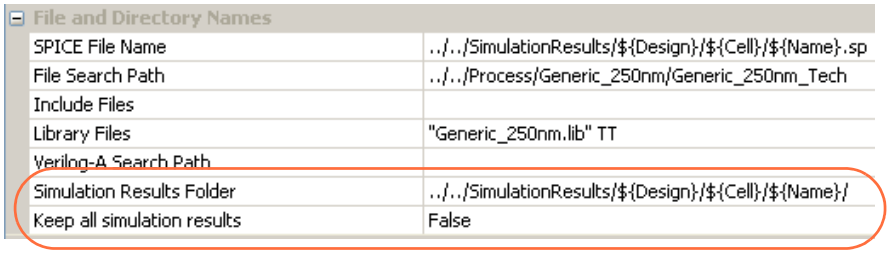
You can easily save or retrieve SPICE simulation settings, or any other type of cell property, from the Properties navigator in TCL file format using either **Copy property** or **Copy tree**. (See “Copying and Pasting Properties” on page 154.)

Note that if the **Simulation Results Folder** field is left empty, output files are considered temporary and are deleted at the end of the S-Edit session. (The default location for output files is the directory indicated in the **SPICE File Name** field.)

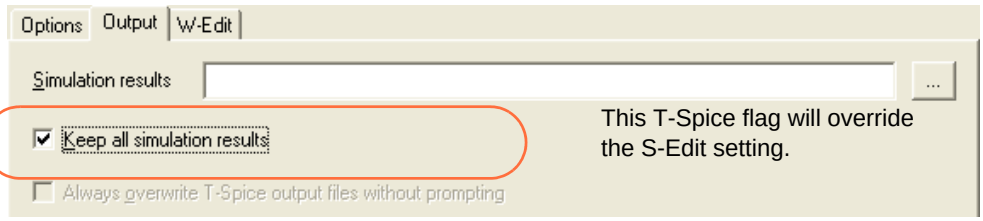
SPICE settings from S-Edit or T-Spice

Note that both S-Edit and T-Spice have fields that control the folder in which simulation results are saved and whether or not simulation results are overwritten.

If there is a value in these fields in S-Edit **Setup > SPICE simulation**, and the simulation is initiated from S-Edit, the S-Edit settings are used.

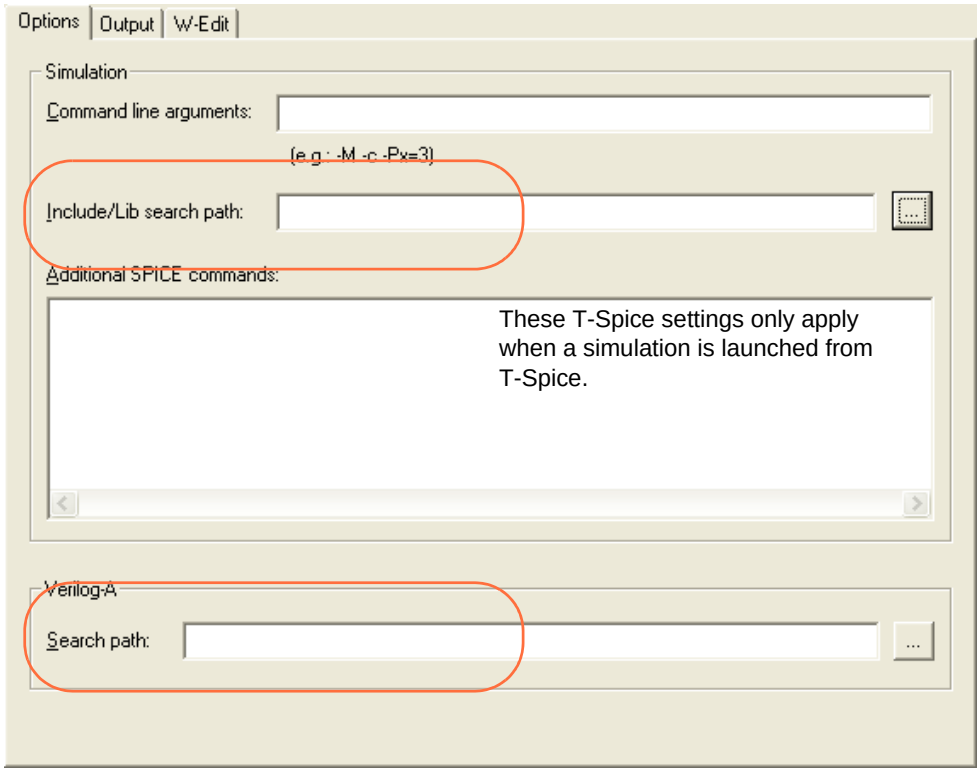


However, if you run a simulation from T-Spice and the **Keep all simulation results** option is checked, the T-Spice setting will override the S-Edit setting.



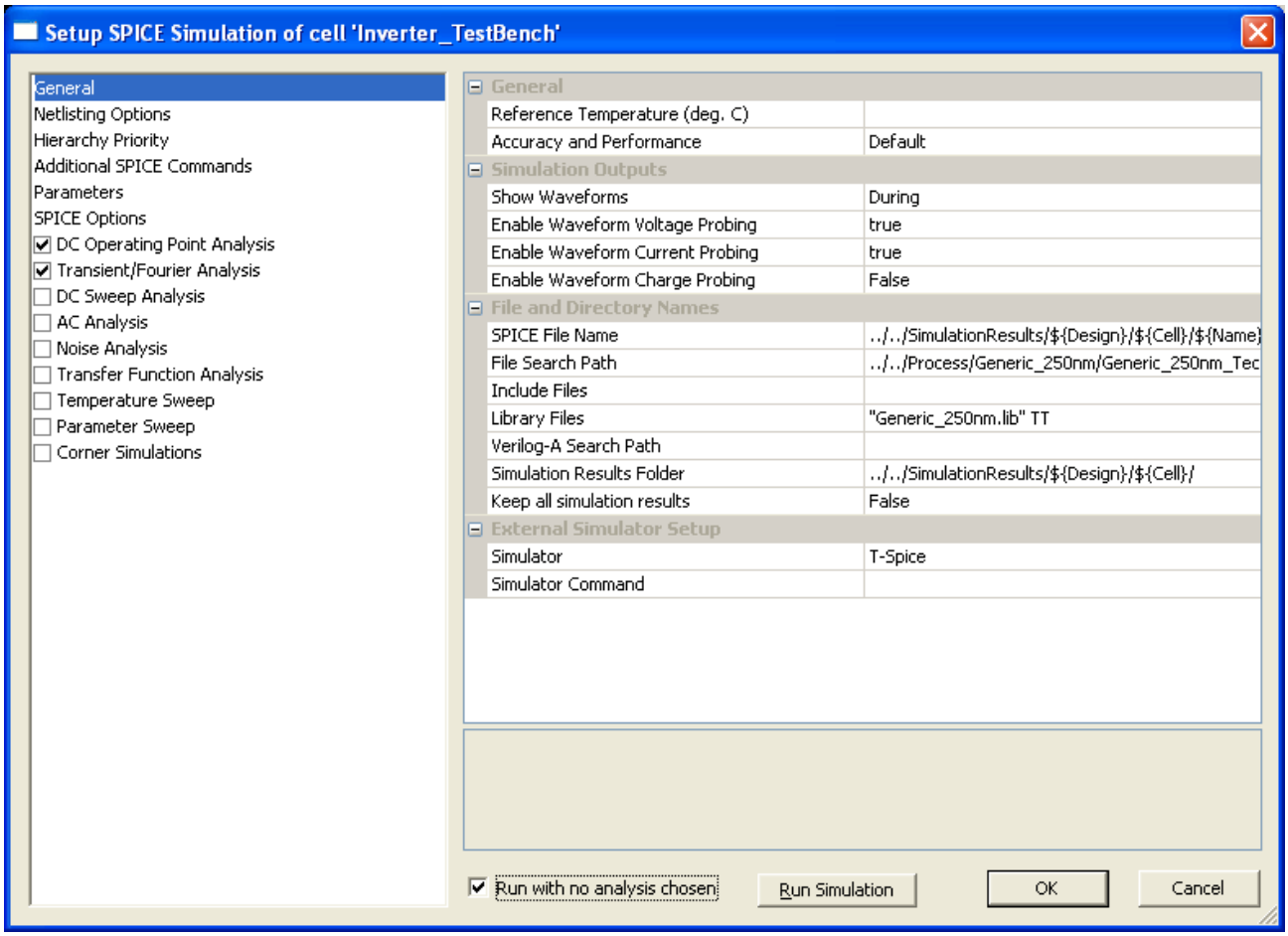
Though both S-Edit and T-Spice have fields for the library, include and Verilog-A files, those in S-Edit are written to the netlist, whereas those in T-Spice are written to a file header.sp that is called by the netlist.

To use the T-Spice settings in **Simulation > Simulation Settings** for include, library or Verilog-A search paths, you need to run the simulation from T-Spice.



General Settings

Note: When a File Search Path is used, the include file name needs to be in double-quotes.



Reference Temperature (deg.C)

Tnom - the nominal or reference temperature at which device model parameters were measured; value may be overridden in individual model definitions using the **tnom/tref** model parameter. Tnom will also represent the default operating temperature of the circuit. (degrees Celsius)
Related T-Spice command: **.option tnom=*temp***

Accuracy and Performance

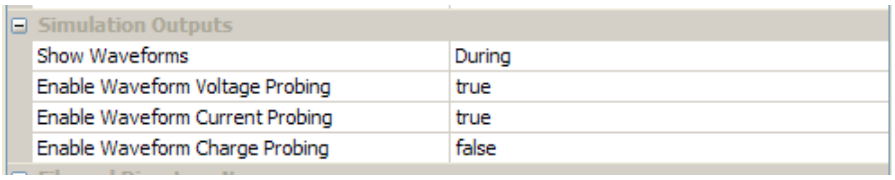
Use this setting to control the trade off between accuracy and performance, where the choices in order — **fast, default**, **accurate**, and **precise** — increase in accuracy and decrease in performance.

The **fast** setting will provide faster simulations with some compromise of accuracy. The **default** setting, which is generally the best setting, balances accuracy and speed. The **accurate** setting will enhance accuracy at some expense of performance. The **precise** setting maximizes accuracy with minimal performance improvements.
Related T-Spice commands: **.option fast** and **.option accurate**

Run with no analysis selected

When checked, allows simulation to proceed when no analysis is selected in the dialog, so that you can use analysis commands embedded in your schematics or netlist. If no embedded simulation commands are found in the input file, S-Edit provides a warning.

Simulation Outputs



Show Waveforms

Select when W-Edit should display the simulation results.
Choices:
During - Display solutions during simulation with realtime updating
After - Display solutions at the end of the simulation
Don't Show - W-Edit will not be activated at all

Enable Waveform Voltage Probing

Indicate whether node voltage values should be included in the probe data output file.
Related T-Spice command: **.option probev=true | false**

Enable Waveform Current Probing

Indicate whether device terminal current values should be included in the probe data output file.
Related T-Spice command: **.option probei=true | false**
Warning: May cause excessively large output files.

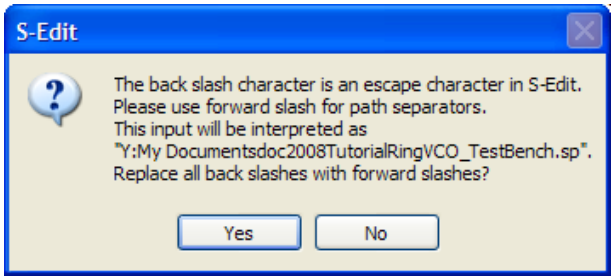
Enable Waveform Charge Probing

Indicate whether device terminal charge values should be included in the probe data output file.
Related T-Spice command: **.option probeq=true | false**
Warning: May cause excessively large output files.

File and Directory Names

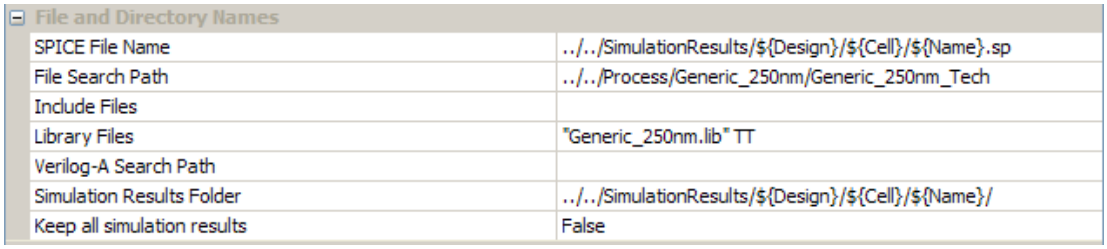
Note: To prevent simulation errors, S-Edit performs syntax checking in the **File and Directory Names** fields of the **Setup SPICE > General** dialog. The backslash character (****) is an escape-sequence prefix in S-Edit. Since these fields are evaluated, you should use forward slashes in any file or path name field.

S-Edit prompts you to accept an automatic fix if it finds the backslash character, as shown below.


S-Edit
The back slash character is an escape character in S-Edit.
Please use forward slash for path separators.
This input will be interpreted as
"Y:My Documentsdoc2008TutorialRingVCO_TestBench.sp".
Replace all back slashes with forward slashes?
Yes
No

Note that if the netlist name is not user-set in S-Edit in the **Spice File Name** field, but a **Simulation Results Folder** is set, the netlist is by default stored in the folder specified in the **Simulation Results Folder** field.

Also, an output directory cannot be created if there is a period (**.**) in the **Simulation Results Folder** name.



File and Directory Names	
SPICE File Name	../../SimulationResults/${Design}/${Cell}/${Name}.sp
File Search Path	../../Process/Generic_250nm/Generic_250nm_Tech
Include Files	
Library Files	"Generic_250nm.lib" TT
Verilog-A Search Path	
Simulation Results Folder	../../SimulationResults/${Design}/${Cell}/${Name}/
Keep all simulation results	False

SPICE File Name

The name of the SPICE file which will be created and simulated when the **Simulate** command is executed.

When the SPICE netlist is generated, all directory and filenames entered in the **General Settings** dialog will be expanded to refer to fully qualified path names. If this is not the desired behavior, then the name should be entered within quotes.

Since these fields are evaluated, you can enter $Cell.out, for example, to output “cellname.out”.

For example, an include file identified as either **./mosis2u.md** or **mosis2u.md** will be written to the SPICE file as ***designpath*/mosis2u.md**, where *designpath* is the directory containing the schematic design. If the filename had been entered as “mosis2u.md”, then it would not have any pathname expansion performed.

Note also that unless a location for the output files (ex. .dat and .out) is specified in this field or the **Simulation Results Folder** field, the default is to consider them temporary in which case they are deleted at the end of the S-Edit session.

File Search Path

Enter the directories that S-Edit will search for library and include files. Use semicolons to separate multiple entries. Relative pathnames are expanded to be absolute unless enclosed in quotes.

Include Files

List files whose contents should be included in the netlist. Use semicolons to separate multiple entries.

Library Files

A list of semicolon separated model library files and optional section names. Note - *libraryname* should be quoted if *sectionname* is included.

Library files may be specified in the same three ways as include files. However, with library files there is also the name of the library inside the file that must be specified. The name of the library is placed immediately after the library name.

“LibfileA” LibnameA; “LibfileB” libnameB; ...

This generates SPICE .lib statements that looks like:

.lib “LibfileA” libnameA
.lib “LibfileB” libnameB

Related T-Spice commands:
.lib *libraryname*
.lib *libraryname sectionname*

Verilog-A Search Path

Enter or browse to the directories containing Verilog-A model files. Separate multiple list entries with semicolons.

Simulation Results Folder

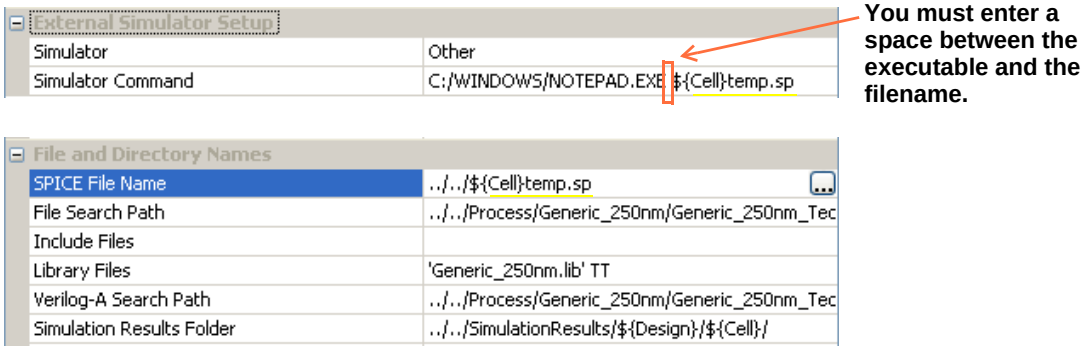
The name of the output file folder which will contain T-Spice simulation results, and can be plotted by W-Edit.

Note: Temporary output files are removed at the end of the S-Edit session unless you enter a directory location in this field or **True** in the **Keep all simulation results** field.

Keep all simulation results

When set to **True**, S-Edit appends a date and time stamp to the Simulation Results folder so that each simulation result is saved to a unique folder and will not be over-written.

External Simulator Setup

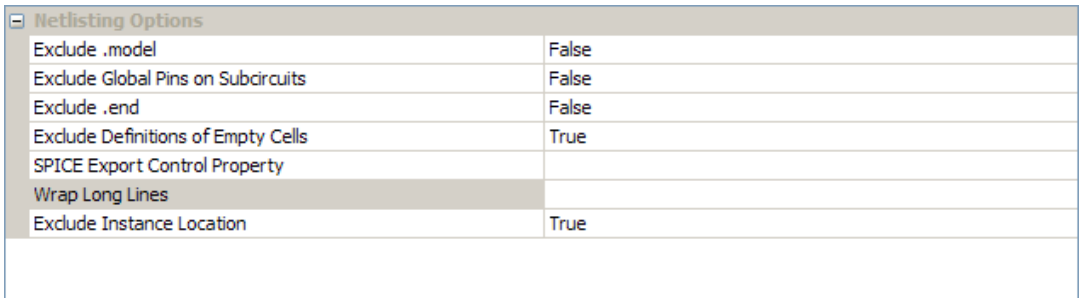


Simulator — Allows you to select a simulation application other than T-Spice.
Choices: **T-Spice, other**

Simulator Command — The command or executable name to launch a SPICE simulator other than T-Spice, and an output filename, which must match the output file entered in the **SPICE File Name** field.

Note: As highlighted above, a space is required between the executable and the filename**.**

Netlisting Options



Exclude .model — When set to True, .model statements are excluded from the netlist (for example, if you plan to include the netlist file in other SPICE netlists.)

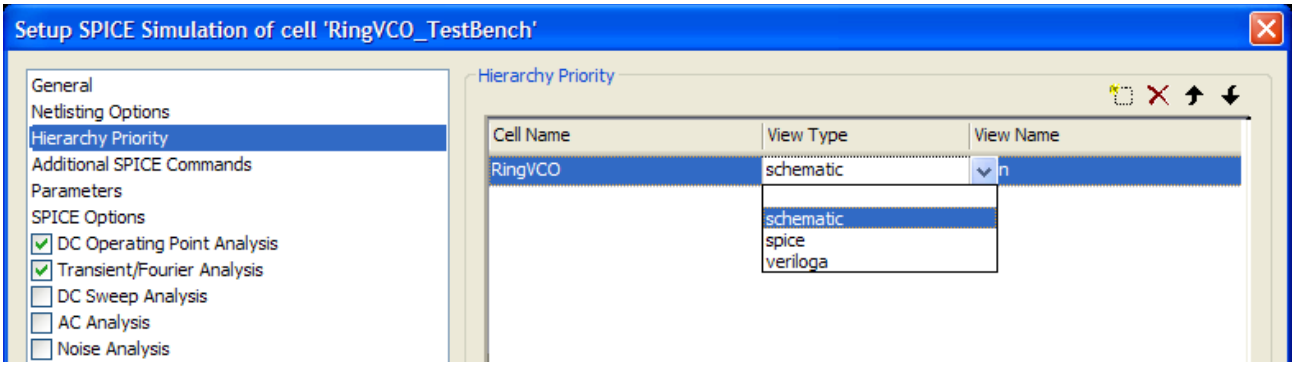
Exclude Global Pins on Subcircuits — When set to True, global ports are not expanded in subcircuit definitions and calls; instead a .global statement is added to the netlist.

Exclude .end — When set to True, suppresses the **.end** statement in the SPICE output file.

Exclude Definitions of Empty Cells	When set to True, excludes definitions of empty subcircuits. The priority for writing subcircuit definitions is as follows: 1. Use the SPICE.DEFINITION 2. If the cell is not empty, write a definition. 3. If the cell is empty (i.e. there is no schematic view or the view contains no instances), write an empty subcircuit definition.
SPICE Export Control Property	Prefix used to control netlisting of devices and subcircuits when running simulations and exporting directly to T-Spice. If left blank, the default is “SPICE”.
Wrap Long Lines	Wrap long lines to specified number of characters. Negative or zero values will be ignored.
Exclude Instance Location	When checked, does not write the instance location coordinates that are prefaced by a dollar sign to the netlist (to prevent errors in tools such as Eldo.)

Hierarchy Priority

Use the **Hierarchy Priority** list in **Setup > SPICE Simulation** to specify which view type S-Edit uses to netlist a cell. This priority also controls view type in the design area.

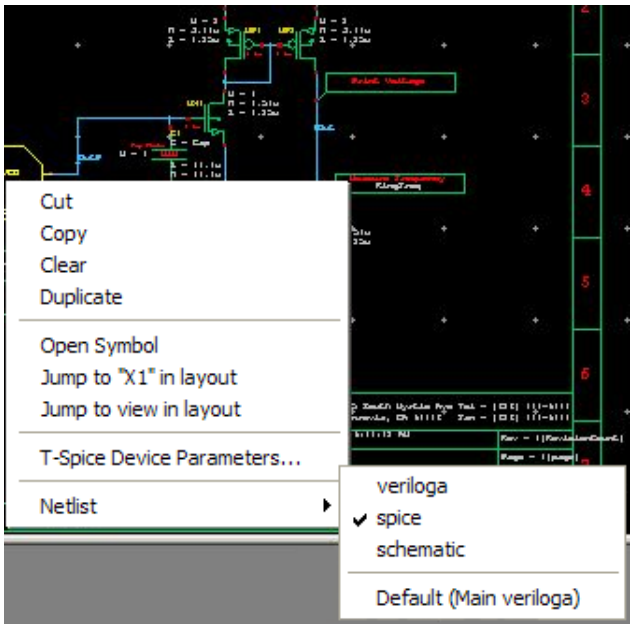


Cell Name Enter the cell name.

View Type Choose from schematic, SPICE or Verilog-A (the default order) representation of the cell to export for simulation.

View Name Specify which, if there are multiples, of the cell instances to use (ex. Voltagesource DC, Voltagesource PWL, etc.).

You can also control this setting from any schematic page using the context menu for instances. Select an instance and use a *slow-right-click* to open the context menu. ("Slow-right-click" means you hold the right mouse button down longer than normal before you release it.)



Note that this slow-right-click option only sets hierarchy priority in the **Setup > SPICE Simulation** dialog — it does not enter a setting for exporting SPICE.

The **Netlist** option contains a list of all the views with the same interface and their types, and a **Default**

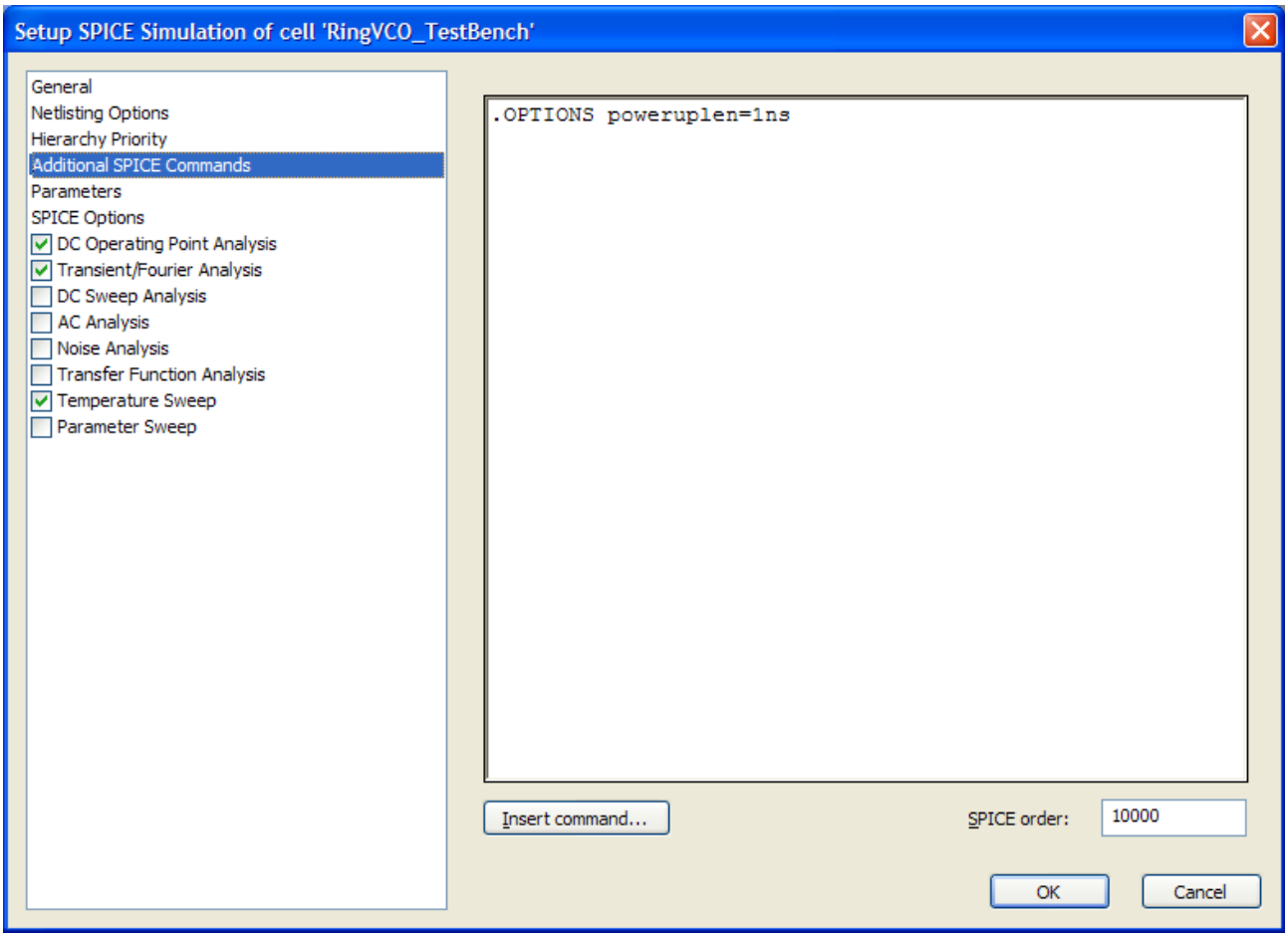
(*view*, *type*). The view with the checkmark is the one that will be used to netlist. Clicking on one of the other views adds the corresponding triplet of values – cell name, view type and view name – to the top of the Hierarchy Priority string, causing that view to be netlisted.

If the view to be netlisted is already present as a triplet in the Hierarchy Priority dialog, the Default item is enabled. If the view to be netlisted does not come from a triplet, the Default item is disabled. Note that if all the views have the same name, or have the same type, the in-common field is not displayed in the list of views under **Netlist**.

Hierarchy Priority also determines the view type in the design area, so that, for instance if the **View Type** in the **Hierarchy Priority** is set to veriloga and you double-click on an instance, the instance will open in the veriloga view.

Additional SPICE Commands

The contents of the "**Additional SPICE commands** field are written verbatim to the SPICE file.



By default, instances are written out according to their SPICE.ORDER setting followed by the simulation commands.

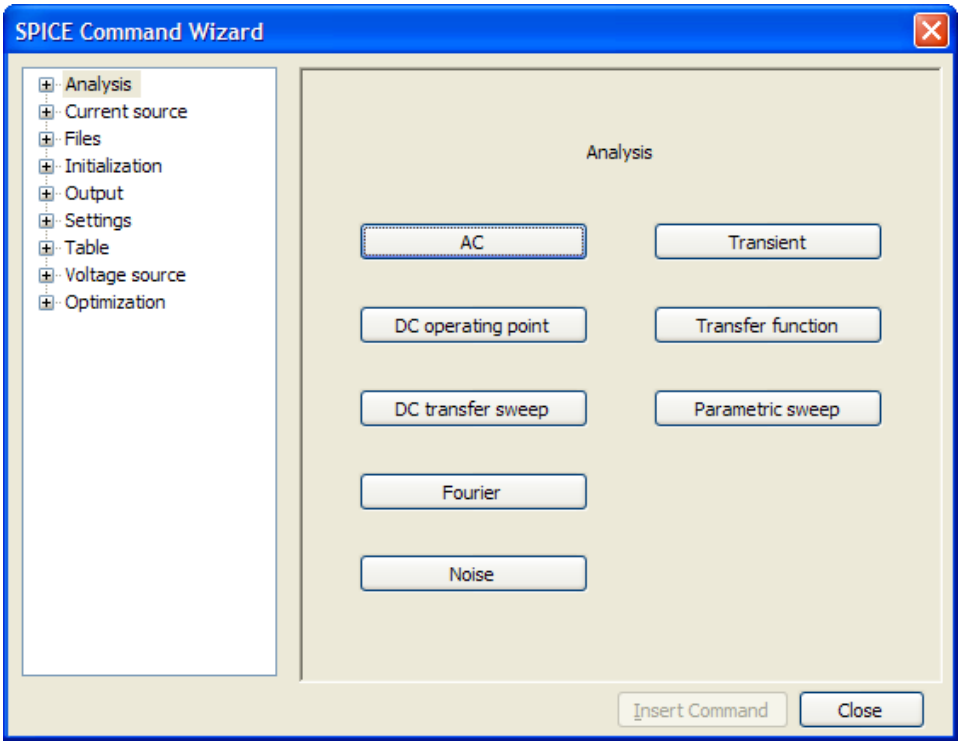
Insert command — Launches the **SPICE Command Wizard** (see "SPICE Command Wizard," below).

SPICE order — Enter the appropriate value in this field to control the position of a command in the SPICE file (for example, to create symbols that use .ALTER, which would need to go after the simulation commands). The default **Spice order** value of 10000 is usually high enough to add simulation commands just before the **.end** statement.

Warning: As with all other S-Edit fields, any single backslash character in a path name will be removed without warning—you should use either forward slashes or double backslashes in the path names you enter.

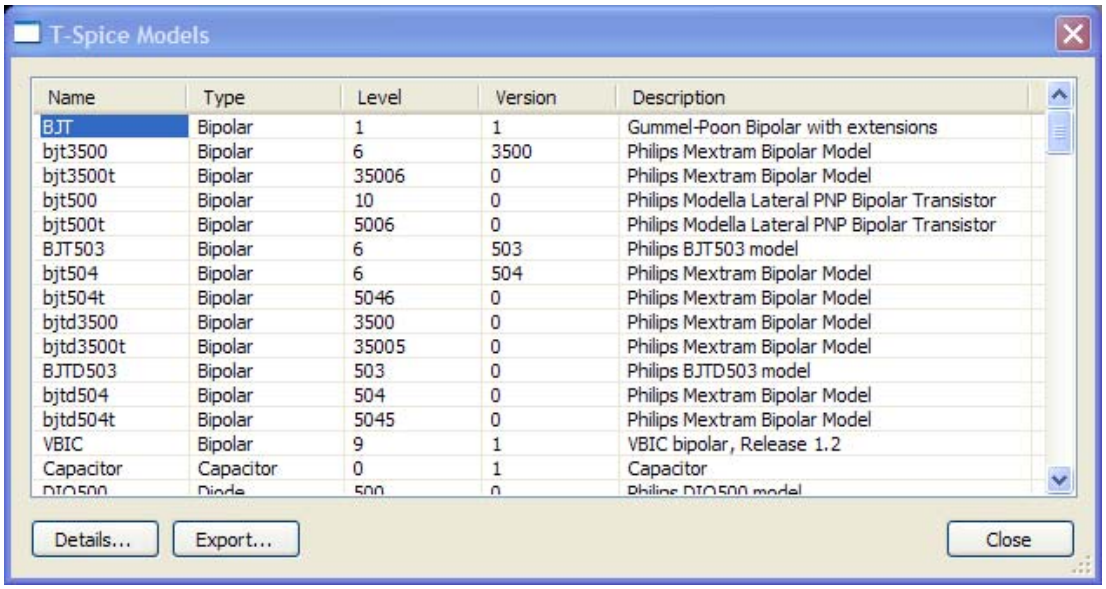
SPICE Command Wizard

The **SPICE Command Wizard** (also available in Tanner's T-Spice) prompts you to enter the correct options and parameters so you can quickly compose SPICE simulation commands with perfect syntax. Commands are grouped by analysis type.

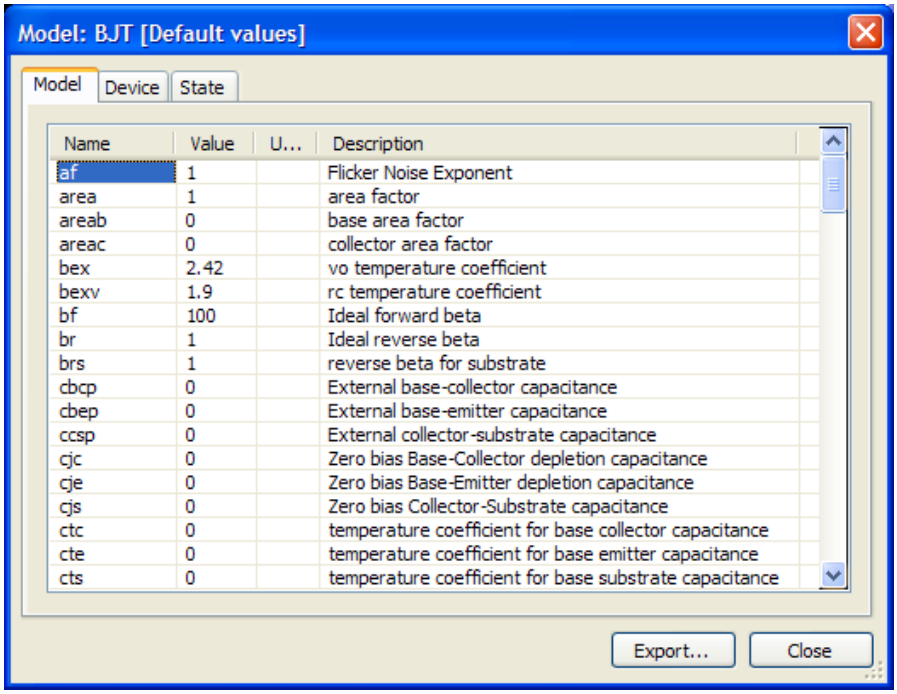


Online SPICE Model and Parameter Lists

The S-Edit help system provides a list of the SPICE model equations the product supports, accessible from **Help > Models Supported by T-Spice**.

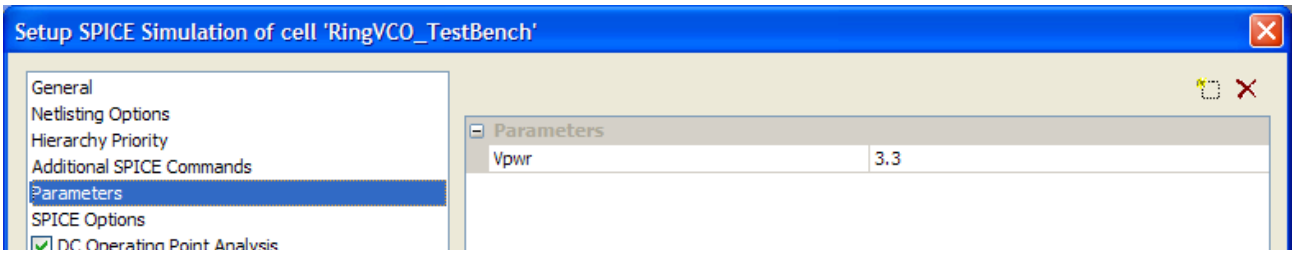


You can sort by each of the columns (**Name**, **Type**, **Level**, **Version** and **Description**) and for each of the models, you can view the supported parameters, devices and states along with their default value and units (as shown below for BJT level 1).



Use the **Export** button on either of these dialogs to export the contents of the current tab to a comma-delimited text file.

Parameters

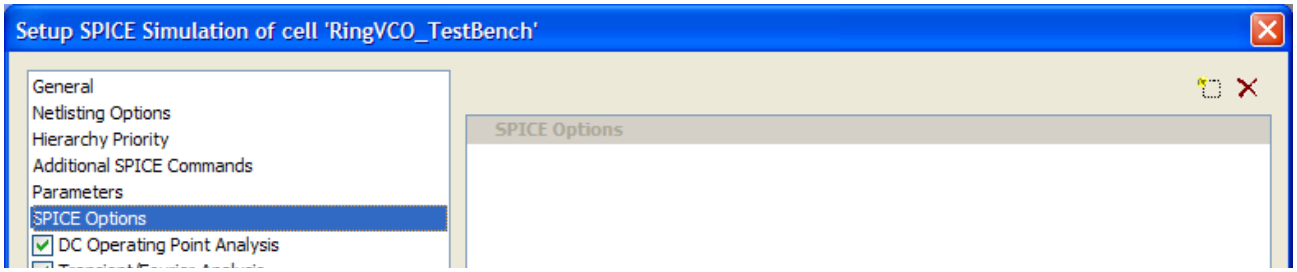


The **Parameter** setup allows you to enter an arbitrarily long list of parameter assignments. Parameters are ***name=value*** pairs which define numeric variables that can be used throughout the SPICE input files, in equations or as model and device parameter values.

Related T-Spice *command*: **.param *name=value***

Use the **create new** button ( ) to enter a parameter and the **delete** button ( ) to remove one.

SPICE Options

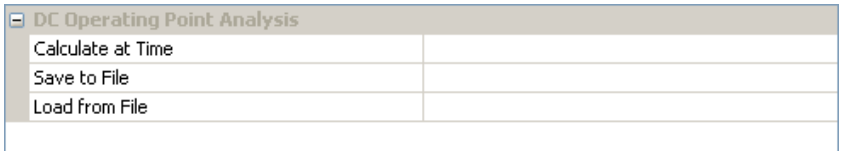


SPICE options are also entered as name and value assignment pairs. The name can be any valid T-Spice option name, and the value is the numeric or string value of the option.

Related T-Spice command: **.option *optionname=setting***

Use the **create new** button ( ) to enter a parameter and the **delete** button ( ) to remove one.

DC Operating Point Analysis



Print DC Operating Point — Indicate whether the DC Operating Point bias and AC small-signal information should be written to the output file.
Related T-Spice command: **.op**

Calculate at Time — A list of transient timepoints at which the bias information and AC small-signal data should be printed.
Related T-Spice command: **.op *T0 T1 T2***

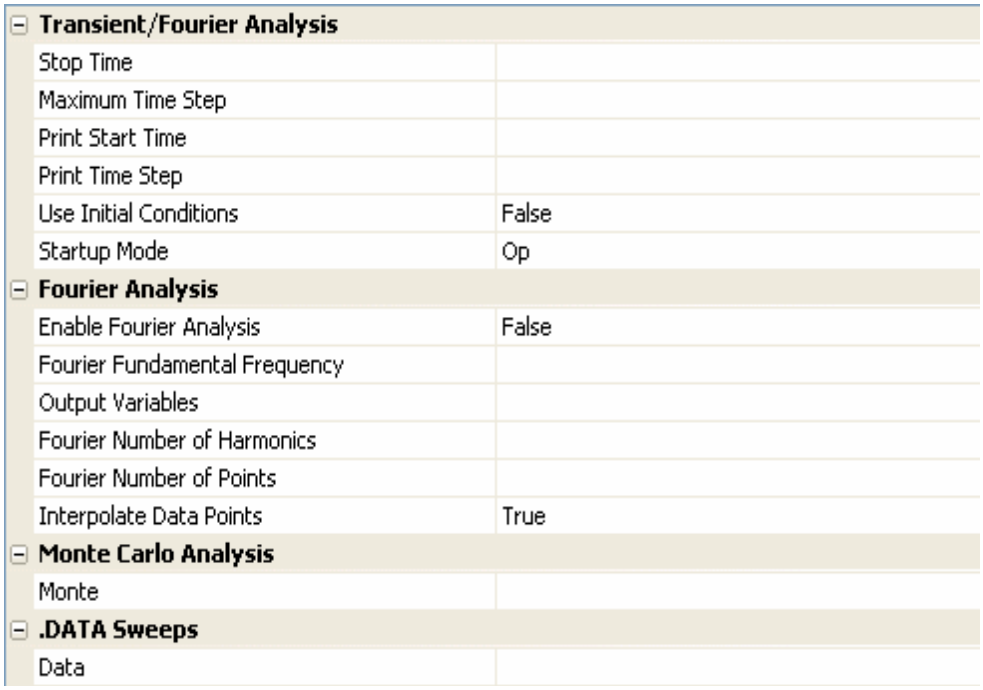
Save to File — Save the node voltage values for the DC operating point to a file in a format which can be loaded in subsequent simulations. Computational time may thus be reduced by providing an improved initial guess for voltages in the subsequent runs.
Related T-Spice command: **.save file=*filename***

Load from File — Load the bias point information from a previous simulation, and use the node voltage values as the initial guess in the DC operating point computation.
Related T-Spice command: **.load file=*filename***

Transient/Fourier Analysis

Fourier analysis can be performed in W-Edit, as well as in T-Spice. There are a number of advantages to performing the Fourier analysis in W-Edit:

- Sampling density can be adjusted
- Start and stop times can be varied
- Windows can be applied in order to reduce the error effects of finite time sampling
- Results can be computed and plotted interactively, without rerunning the simulation



Transient/Fourier Analysis	
Stop Time	
Maximum Time Step	
Print Start Time	
Print Time Step	
Use Initial Conditions	False
Startup Mode	Op
Fourier Analysis	
Enable Fourier Analysis	False
Fourier Fundamental Frequency	
Output Variables	
Fourier Number of Harmonics	
Fourier Number of Points	
Interpolate Data Points	True
Monte Carlo Analysis	
Monte	
.DATA Sweeps	
Data	

Stop Time
The transient simulation stop time, i.e. the total simulation time.
Related T-Spice command: **.tran *Tstep Tstop***

Maximum Time Step
The maximum timestep to be taken during the transient simulation.
Related T-Spice command: **.tran *Tstep Tstop***

Print Start Time
The time when simulation results printout will begin. If not set, then the results printout will begin immediately, at time = 0.
Related T-Spice command: **.tran *Tstep Tstop* START=*Tstart***

Print Time Step
The time increment at which solutions are printed.
Related T-Spice command: **.option prtdel=*step***

Use Initial Conditions
Instructs the simulator to use node voltage initial condition values at startup time 0, rather than computing the DC operating point.
Related T-Spice command: **.tran *Tstep Tstop* UIC**

Startup Mode
Select the transient simulation startup mode.
Choices:
OP - standard DC operating point computation (default)
Powerup - voltage and current sources are ramped up
Preview - Does not perform an actual transient analysis; for previewing source values.
Related T-Spice command: **.tran/powerup** and **.tran/preview**

Enable Fourier Analysis
True or False.

Fourier Fundamental Frequency
The fundamental frequency about which the harmonic components are computed.
Related T-Spice command: **.four *freq outputs***

Output Variables
List of voltage and current values for computation of spectral components.
Related T-Spice command: **.four *freq outputs***

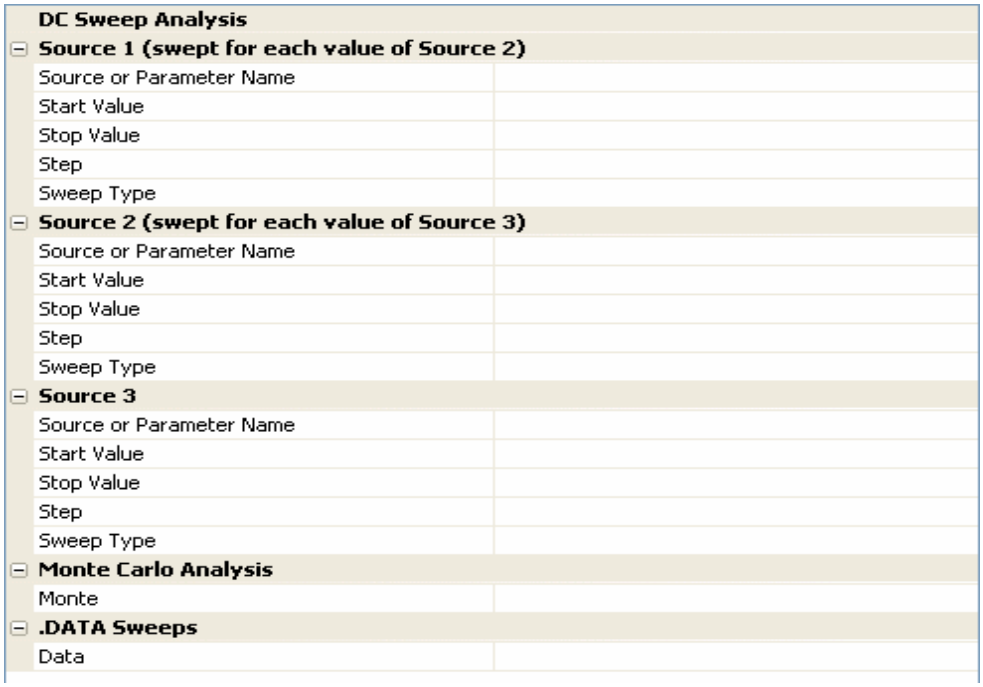
Fourier Number of Harmonics	The number of harmonic frequencies for which Fourier components are computed. *Related T-Spice command:* **.four *freq outputs* nfreqs=*N***
Fourier Number of Points	The number of points over which transient analysis data is interpolated to fit. *Related T-Spice command:* **.four *freq outputs* npoints=P**
Interpolate Data Points	If 0, T-Spice inserts an actual computed time point at each place where a Fourier analysis time point is needed without interpolating transient data to fit on *np*. If 1, Fourier analysis is based on interpolated data. (Default: 1.) *Related T-Spice command:* **.four *freq outputs* interpolate=I**
Monte	Enter the number of Monte Carlo simulations to perform.
Data	Names of .DATA statements that contain sweeps

DC Sweep Analysis (or DC Transfer)

The **DC Sweep Analysis** dialog permits the user to define up to three levels of sweeping for DC analysis. DC sweeping is also referred to as DC transfer analysis. This analysis computes the DC states of a circuit while a voltage or current source is swept over a given interval. T-Spice allows three or more levels of source sweeping, in which the first source sweep is the innermost loop, and is swept for each value of the second source, which in turn is swept for each value of the third source.

In addition to sweeping current and voltage source values, parameter values may be swept in order to yield a DC curve as a function of the parameter value. Parameters are defined in the **Parameters** section

of the setup dialog, and also include the intrinsic **temp** temperature value, which is the operating temperature of the circuit.



Source 1 (Swept for each value of Source 2):

Source or Parameter Name	Voltage source, current source, or parameter name to sweep.
Start Value	The beginning value of the sweep variable.
Stop Value	The final value of the sweep variable.
Step	The value step size for linear sweeps, or the number of points per decade/octave for decade and octave sweeps.
Sweep Type	The type of sweep and *associated T-Spice commands*: ▪ linear (**lin**) - *.DC variable linear start stop increment* ▪ logarithmic decade (**dec**) - *.DC variable dec start stop increment* ▪ logarithmic octave (**oct**) - *.DC variable oct start stop increment* ▪ **list -** list of values in the start value field ▪ <**disabled**> - use to disable a parameter sweep without having to delete values.

Source 2 (Swept for each value of source 3):

Source or Parameter Name	Voltage source, current source, or a parameter name to sweep.
Start Value	The beginning value of the sweep variable.
Stop Value	The final value of the sweep variable.
Step	The value step size for linear sweeps, or the number of points per decade/octave for decade and octave sweeps.

Sweep Type The type of sweep.

Source 3:

Source or Parameter Name Voltage source, current source, or a parameter name to sweep.

Start Value The beginning value of the sweep variable.

Stop Value The final value of the sweep variable.

Step The value step size for linear sweeps, or the number of points per decade/octave for decade and octave sweeps.

Sweep Type The type of sweep.

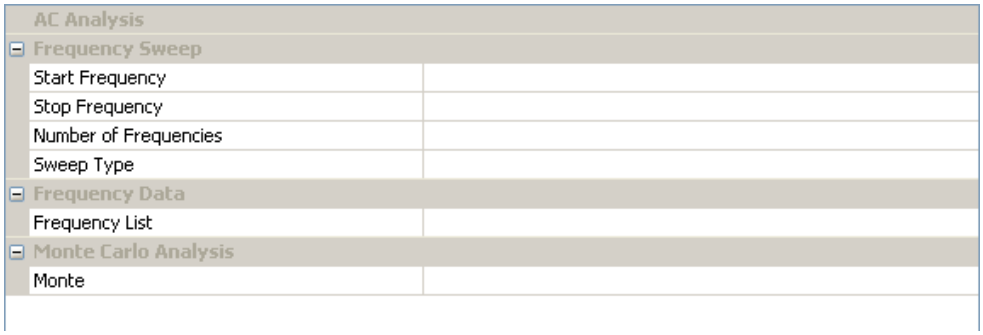
Monte Carlo Analysis

Monte Enter the number of Monte Carlo simulations to perform.

.DATA Sweeps

Data Names of .DATA statements that contain sweeps

AC Analysis



Start Frequency The beginning frequency.

Stop Frequency The final frequency.

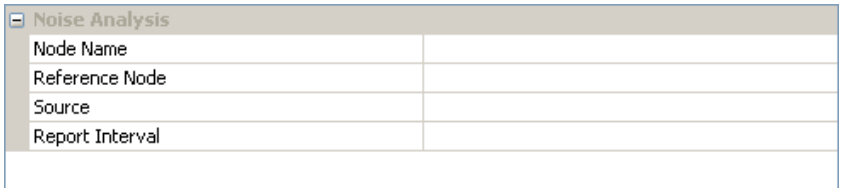
Number of Frequencies The number of frequency steps - total steps for linear sweeps, or steps per decade/octave for logarithmic sweeps.

Sweep Type The type of sweep.
Choices and *Associated T-Spice commands*:
linear (lin) - **.AC linear *Fstart Fstop Fstep***
decade (dec) - **.AC dec *NF Fstart Fstop***
linear (lin) - **.AC oct *NF Fstart Fstop***

Frequency List Specify a list of frequencies (values) for which the analysis is to be performed.

Monte Enter the number of Monte Carlo simulations to perform.

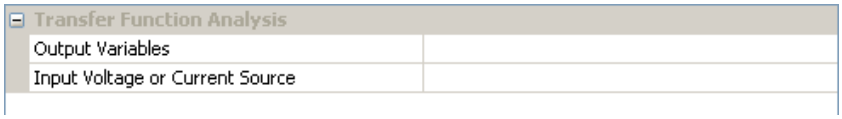
Noise Analysis



Node Name	Node at which the total noise is to be computed.
Reference Node	Reference node for the nodal voltage (default: GND).
Source	Voltage or current source to which input noise is referred.
Report Interval	Interval for printing noise report summaries; i.e. print a report every **N** frequencies.

Related T-Spice command: **.noise V(*node*, *ref*) *source interval***

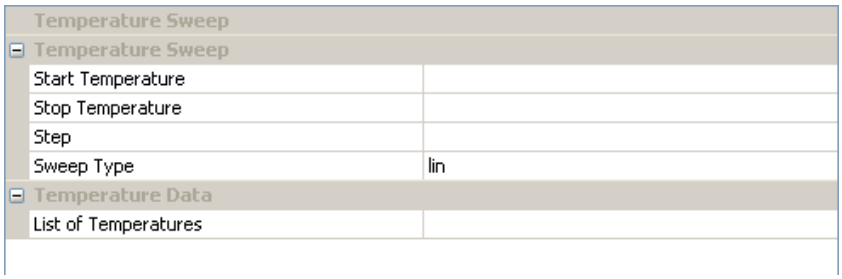
Transfer Function Analysis



Output Variables	The output small-signal variables. This is a list of any valid **.print** commands; e.g. v(out) i1(mos2)
Input Voltage or Current Source	The small-signal input source; i.e. the name of a current or voltage source in the schematic.

Related T-Spice command: **.tf *outvar invar***

Temperature Sweep



Start Temperature — The beginning operating temperature sweep value (degrees Celsius)

Stop Temperature — The final operating termperature value

Step Size or # of Steps — The temperature step size for linear sweeps, or the number of points per decade/octave for logarithmic sweeps

Sweep Type — The type of sweep.
Choices and *Associated T-Spice commands*:
linear (lin) - **.step temp linear *Tstart Tstop Tstep***
decade (dec) - **.step temp dec *NT Tstart Tstop***
linear (lin) - **.step temp oct *NT Tstart Tstop***

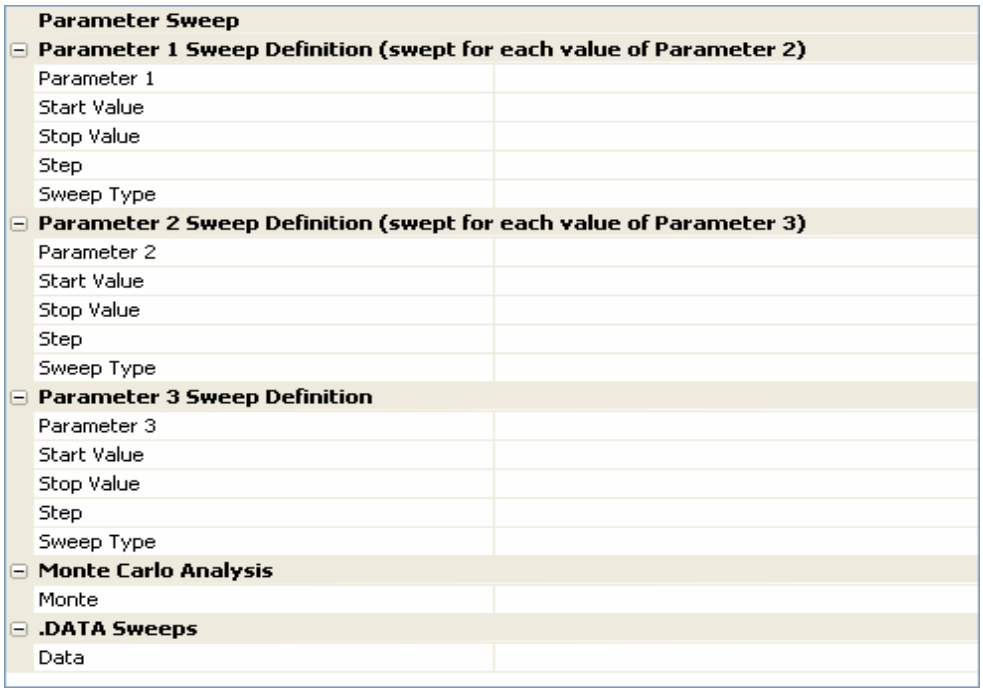
List of Temperatures — A list of circuit operating temperatures. All analyses will be rerun for each temperature.
(degrees Celsius)
Related T-Spice command: **.temp *T0 T1 T2* ...**

Note: The temperature sweep as defined by the four other variables and the **List of Temperatures** entry will generate two separate temperature variation commands. Typically only one or the other type of temperature variation will be used, not both.
Since every type of analysis in the SPICE netlist will be rerun for each temperature point, temperature sweeps should be used judiciously.

Parameter Sweep

The **Parameter Sweep** analysis dialog permits the user to define up to three levels of parametric sweeping. For each parameter value of the sweep, *every analysis command* which has been defined will

be performed - i.e. for a sweep of length N, there will be N total transient, DC, and AC simulations performed.



Parameter 1 Sweep Definition (Swept for each value of Parameter 2):

Parameter Name The first sweep parameter.

Start Value The beginning value of the sweep variable.

Stop Value The final value of the sweep variable.

Step The value step size for linear sweeps, or the number of points per decade/octave for decade and octave sweeps.

Sweep Type The type of sweep and *associated T-Spice commands*:

- linear (**lin**) - *.step variable linear start stop increment*
- logarithmic decade (**dec**) - *.step variable dec start stop increment*
- logarithmic octave (**oct**) - *.step variable oct start stop increment*
- **list** - list of values
- <**disabled**> - use to disable a parameter sweep without having to delete values.

Parameter 2 Sweep Definition (Swept for each value of Parameter 3):

Parameter 2 The second parameter to sweep.

Start Value The beginning value of the sweep variable.

Stop Value The final value of the sweep variable.

Step Size or # of Steps	The value step size for linear sweeps, or the number of points per decade/octave for decade and octave sweeps.
Sweep Type	The type of sweep.
Parameter 3 Sweep Definition:	
Parameter 3	The third parameter to sweep.
Start Value	The beginning value of the sweep variable.
Stop Value	The final value of the sweep variable.
Step Size or # of Steps	The value step size for linear sweeps, or the number of points per decade/octave for decade and octave sweeps.
Sweep Type	The type of sweep.
Monte	Enter the number of Monte Carlo simulations to perform.
Data	Names of .DATA statements that contain sweeps

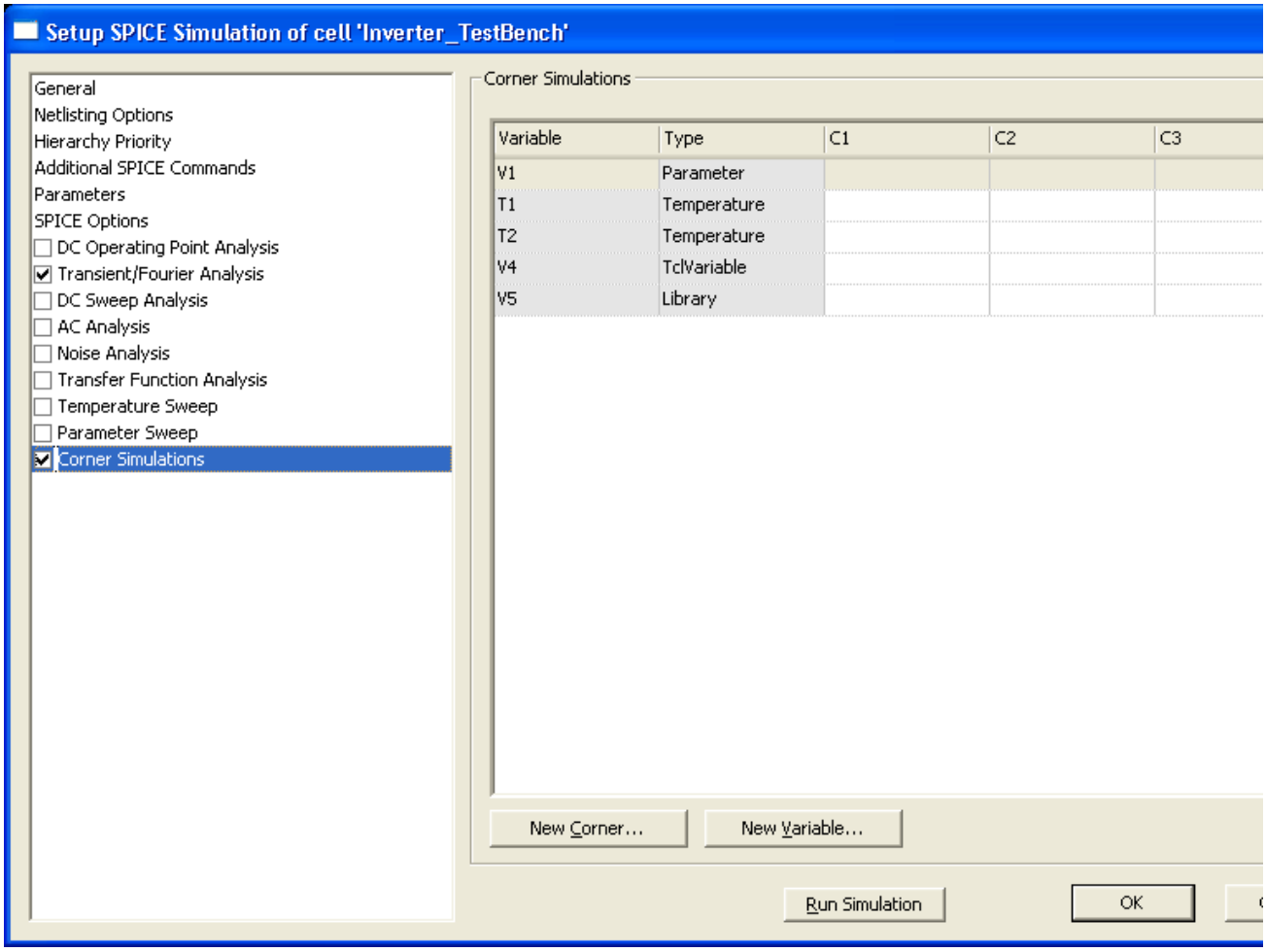
Corner Simulations

Corner simulations let you examine points in the parametric space at which you want to confirm that the circuit operates. Typically one is assessing high/low temperatures, fast/slow device models, high/low power supply voltages, etc.

Corner simulation uses a grid control, with columns that correspond to corners, and rows that correspond to variables. First you create a corner, then assign it variables, then enter values for those variables.

Export SPICE has a checkbox that allows you to **Create separate file for each corner** so that consecutive results are not overwritten.

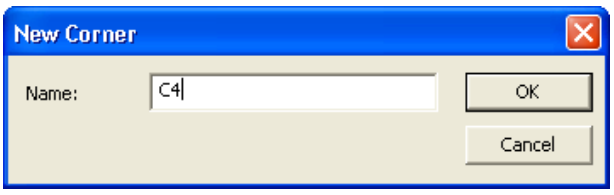
.



New Corner

Each corner is a separate simulation that can have any number and type of variables. Each corner name must be a unique, single word letter or number combination.

Use this button to open the **New Corner** dialog where you can add and name each corner simulation.

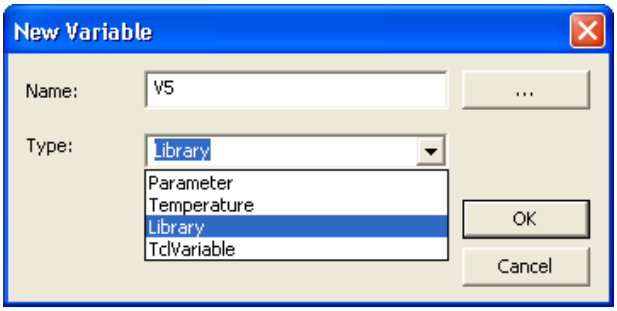


New Variable

There are four possible kinds of variables:

- Parameter
- Temperature
- Library (Default extensions for model files are **.md**, **.mod**, **.l** and **.lib**) Note that when you choose a library variable, the browse button is enabled and will launch a file open window.
- Tcl variable

Use this button to open the **New Corner** dialog where you can add and name each corner simulation.



Once you have added a new variable you can enter its value in the grid.

Running Simulations

Once you have defined the simulations you want to run, use **Tools > Start Simulation** or click on the icon to start SPICE simulation using Tanner's T-Spice application. (Use the **General** page in **Setup SPICE Simulation** to set whether W-Edit should display the results during simulation, after simulation, or not at all.)

If a cell was opened "in context" after clicking on the **Run** (green) button, the top level will be simulated (even if this cell has its own simulation settings). To simulate just a lower-level cell you should open it directly from library browser, as opposed to pushing into it from a higher level of the hierarchy.

Note that the SPICE control property used when running a simulation (**Tools > Start Simulation)** and when exporting directly to T-Spice (**File > Export SPICE**) are independent; they can have entirely different settings that S-Edit does not synchronize.

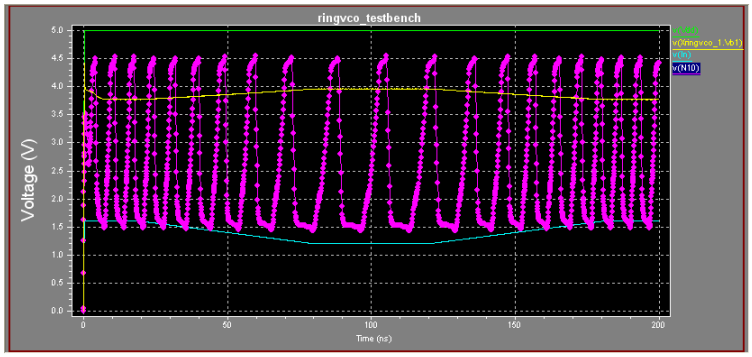
Probing Waveforms

Waveform probing allows you to probe nodes and devices in an S-Edit design to examine their circuit simulation results. The process takes place in three stages:

[1] In the *export* stage, you export a SPICE netlist from your S-Edit design file with at least one type of waveform probing enabled in **Setup > SPICE Simulation > General** so that S-Edit will write **.probe** commands to the netlist.

[2] In the *simulation* stage, you use T-Spice to simulate the exported netlist. When T-Spice encounters a **.probe** command during simulation, it outputs data to the probe data file, a binary file.

[3] In the *probing* stage, you select a net or device for analysis using a probe tool. When you probe the schematic design, S-Edit invokes W-Edit, which automatically displays the waveforms corresponding to the simulation results if you have selected **Plot,** or places the name of the probed device in the waveform calculator if you have selected **Calculator**.



Using the .probe Command

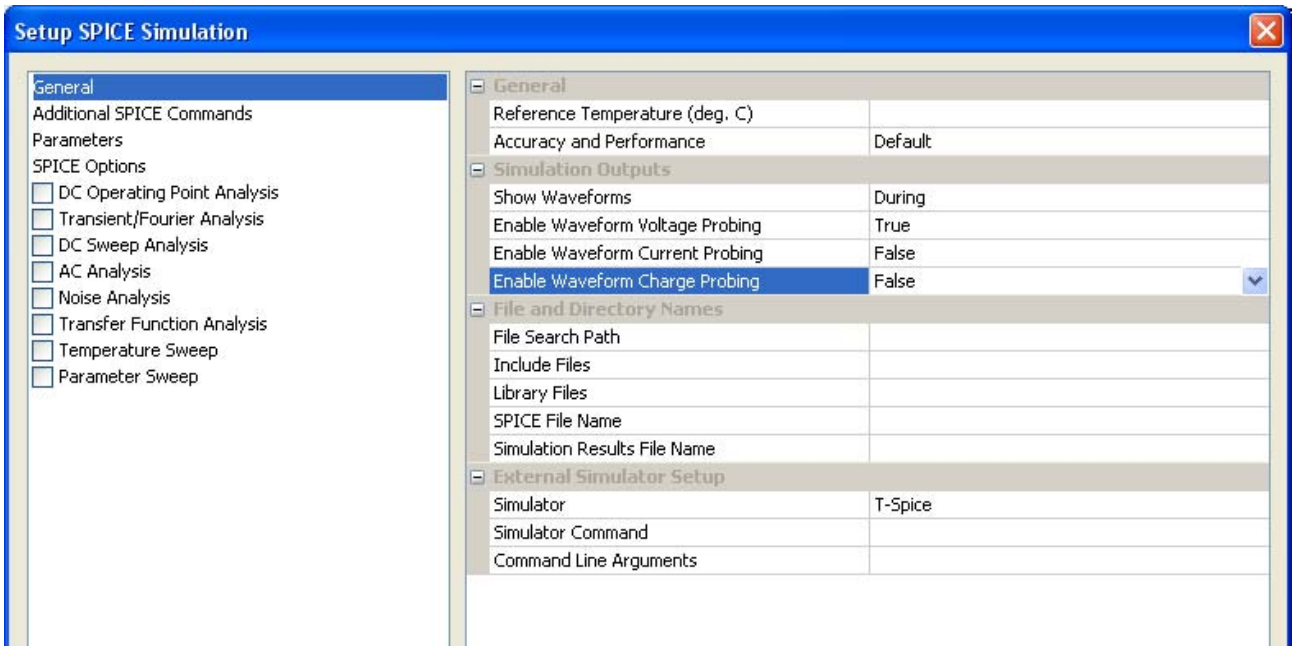
For additional control of waveform probing, you can manually enter a **.probe** command in the SPICE netlist, or from the **Additional SPICE Commands** page of **Setup > SPICE Simulation**.

For example, to cross-probe specified nodes with W-Edit, you can add **`.probe tran V(nodename)`** commands so that only the desired nodes will be output during simulation. Prior to running a simulation you disable all waveform probing, then afterwards enable waveform voltage probing. S-Edit will cross-probe on voltage nodes using the .tsim file to W-Edit.

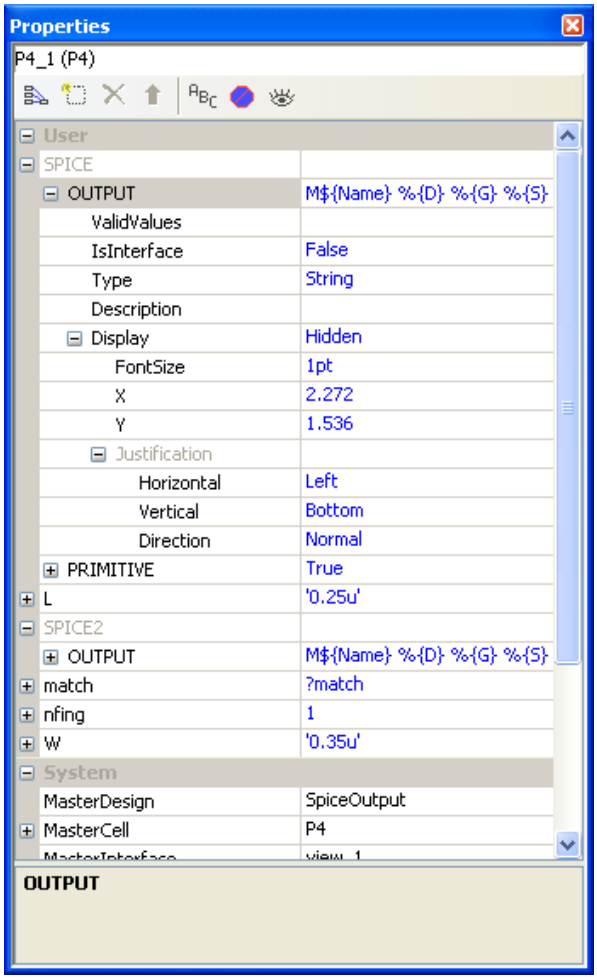
Several T-Spice options pertaining to waveform probing may also be set with the **.options** command. (For more information on the **.options** and **.probe** commands, refer to “Simulation Commands” in the *T-Spice User Guide and Reference*.)

Probing Setup

You must enable at least one of the waveform probing types—voltage, current or charge—for S-Edit to collect probing data during simulation, by setting the appropriate field to “true” on the **General** page of **Setup SPICE Simulation**.



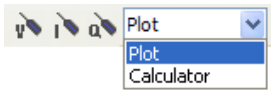
You can also use the Properties navigator to enter these values.



When available, all types of simulation results (transient, AC, etc.) will be displayed in W-Edit for each probe action. If a probe request is made which cannot be fulfilled, W-Edit will indicate that no current data is available.

Probe Tools

The behavior of the probe tool depends on the data included in the simulation and the object being probed. Use **Tools > Probe V** to probe nets, wires, net labels and ports. Use **Tools > Probe I** or **Probe Q** to probe primitive device terminals and subcircuit pins.



All probe tools use a double-click to drill down into the hierarchy. When you reach the level of a primitive, S-Edit displays the small signal outputs.

Differential Probing

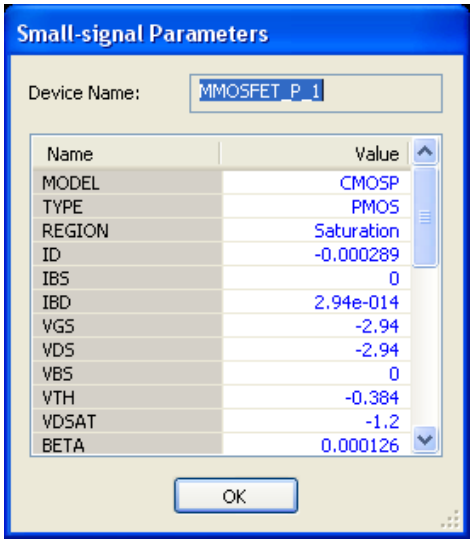
Differential voltage probing may be performed by setting a reference net; then subsequent voltage probes display the voltage difference between the probed net and the reference net. The difference is displayed using an arithmetic trace expression in W-Edit. The reference net is set by selecting a net, then pressing the **Reference** button. This net will remain the reference net until the **Reference** button is pressed again with different net selected. Pessing the **Reference** button with no nets selected cancels the reference net setting, and normal probing will resume.

Probing for Unique Instance Properties—"In Context" Values

You can use the **Push into Context** button, or double-click using a **Probe** tool, to open a specific instance. (This operation will yield no results if a cell is not an instance.)

Depending on the object and type of analysis, when you push to deeper levels of the design hierarchy to probe within the subcircuit you can see, for example, small-signal parameter values, property values derived from expressions, or operating-point voltages.

While editing "in context" you can only select or edit objects contained in the top level instance. You can continue to push down to lower levels within an instance until you reach a SPICE primitive.



Use the **Pop Context** icon to "pop" out of the last instance you pushed into, until you return to the top level of the cell hierarchy.

You can also use the push and pop features to trace nets in your design (see "Net Highlighting" on page 140.)

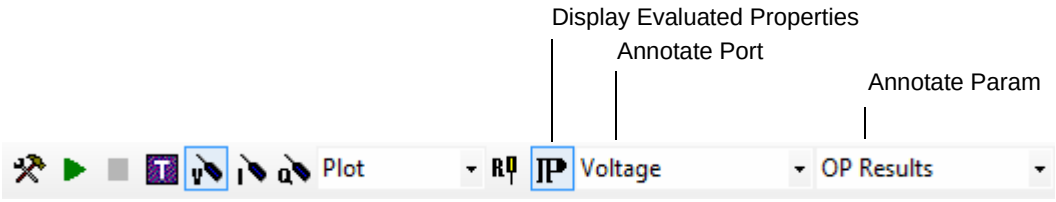
Probing Buses

You can select multiple or even all signals when you probe on a bus for voltage or current probing so the entire bus can be viewed at once instead of as separate different signals.

Displaying Schematic Annotations

A symbol may contain evaluated labels or properties that can be used to display various properties on the schematic. See the chapter **“Evaluated Properties and Labels”** (page 160) for more details.

With **Display Evaluated Properties** turned On, use the **Annotate Port** Drop Down in the Simulation Toolbar to control how Annotate Port Properties or Labels on a symbol are displayed on instances of symbols in a schematic view.



Annotate Port can display the following values

- Port name
- Net name
- DC Voltage
- DC Current
- DC Charge
- No Display

Use the **Annotate Param** Drop Down to control how Annotate Param properties or labels on a symbol are displayed on instances of symbols in a schematic view. This Drop-Down is also used to control the display of cdsParam(n) labels. The Annotate Param Drop-Down can display

- Instance Parameters (as specified in the schematic)
- Operating Point (Small Signal) Parameters
- Model Parameters
- Device Parameters (as calculated in T-Spice)
- No Display

Index

SYMBOLS

?P property reference, 162
@P property reference, 163
%port property reference, 161
$P property reference, 162

A

AC small-signal data, to the output file, 222
accurate,in SPICE simulation, 211
allow editing, 37, 43, 63
analysis
    using embedded from netlist or schematics, 211
ancestor cells, 87
annotating ports, 165
arranging windows, docking, 34
arrays, 127, 130
    example, 130
autocalculate, to set grid size, 40

B

backup files, design, 60, 69
bias point information, 222
boxes, drawing, 101
bundles, 127, 130
buses, 127
    creating, 127
    grouping, 128
    incrementing, 127
    labeling, 127
    prefix notation, 128
    probing, 236
    repeat operator, 128
    suffix notation, 128
    syntax, 127
    two-dimensional, 128
buttons
    adding to toolbars, 13
    creating customized for toolbar, 13
    renaming, 13
    TCL functions for, 13

C

Cadence
    CDF properties, 174
    cdfDump files, 173, 176
    cdsParam properties, 173
    cdsParams, 173
    functions, 162
    property values pPar, iPar, atPar, 174
callbacks, 150, 156
capped nets, 134
capping
    global nodes, 134, 192
cdfDump files, 173, 176
cdsParam properties, 173
cell
    printing a title block, 77, 78
cell hierarchy
    full, 87
    parents and children, 88
    report in text format, 89
cell list, filtering, 86
cells
    ancestor, 87
    child, 87
    copying, 72
    creating, 71, 73
    deleting, 75
    descendant, 87
    implicit connections, 120
    instancing, 74, 86, 115
    interface, 53, 56
    locking, 43
    multiple interfaces, example, 57
    multiple symbols, 58
    naming constraints, 138
    opening, 70, 71, 85
    parent, 86, 87
    previewing, 85, 86
    primitive, 86
    redirected, 67
    renaming, 75
    schematic view, 54
    selecting all instances, 119
    top-level, 86, 196
    unresolved, 65
    validation, 138

view types, 53, 54, 55
circles, drawing, 101
Command window
as a log, 28, 200
excluding messages from, 32
filtering content, 32
for executing scripts, 28
formatting display, 31
toggling display, 28
commands
adding to toolbars, 15
creating customized, 15
renaming, 15
TCL functions for, 15
connection points, 120
connection points, displayed in design window, 120
coordinates
using to edit objects, 106
copying
objects in the design window, 110
work area as bitmap, 112
corner simulations, 230

D

DC operating point
saving node voltage values, 222
DC operating point analysis, 222
loading bias point information, 222
DC operating point analysis, bias, 222
DC transfer, *see DC sweep analysis*
default,in SPICE simulation, 211
deleting objects, 112
descendant cells, 87
deselecting objects, 108
automatic deselection, 108
design, 135
design area, 22
design checking
on export, 135, 180
design checks
on export, 180
design folders, 61
design hierarchy
ancestor cells, 87
child cells, 87
descendant cells, 87
moving through levels with "push" and "pop", 167
parent cells, 87
viewing, 87
design information
displaying, 52
design windows
arranging, 80
changing focus with "backward"/"forward", 81
changing focus with "previous"/"next", 81
previous view, 82
redrawing, 81
reuse behavior, 44, 81
designs
adding, 85
backup files, 60, 69
backup files for, 69
closing, 69
creating new, 60
file structure, 60
locking, 43, 60
opening, 62
previewing print layout, 79
printing, 76
removing, 86
renaming, 69, 70
saving, 60, 69
device terminal charge, 211
device terminal current, 211
devices
replacing, 154
display grid
coordinates, 39
minimum feature size, 39, 40
origin point, 39
showing or hiding, 82
display units
internal units, 39
displaying and hiding, 119, 161
double-click behavior, 44, 70, 87, 98, 99, 120, 167
Draw toolbar, 11
duplicating objects
in the design window, 110

E

EDIF files, 168
exporting, 135, 180, 191
importing, 169
SPICE.OUTPUT, 171, 174
translation for Mentor, 176
translations for Cadence files, 173
translations for ViewDraw files, 170
edit replace, 154
Electrical toolbar, 11
evaluated fields, 61, 211
evaluated properties, 151, 160, 161, 163
prefixes %, $, ? and @, 161
visibility of, 150
example
cell library, 58
filtering the menu list, 10
interface view, 56
interface views, 57

multiple interface views, 59
multiple symbols, 58
view types, 57
exclusive access, 37, 43, 63

F

fast,in SPICE simulation, 211
File > Hierarchy Report, 90
file formats
text files, 49
TPR, 198
file name
evaluation, 61, 211
files
locking, 37, 43, 63
Find navigator, 28, 90
flipping objects, 110
for waveform probing, 233
Fourier analysis, 222
Full hierarchy, 87
fundamental frequency, for Fourier analysis, 223

G

global nets, 132, 133
as defined by global ports, 123
capped, 133
naming, 133
propagating, 133
global nodes, 190, 192
capping, 134
exporting, 190
naming, 133
global ports, 173
creating, 131
naming for global nets, 133
global symbols, 132
collisions, 133
naming, 133
grid
coordinates, 39
minimum feature size, 39, 40
mouse, 39, 40
origin, 39
showing or hiding, 82
snapping, 39, 40

H

harmonic frequencies, in Fourier analysis, 224
help system
for TCL commands, 200
hidden properties, 148
hiding
instances, 117
Hierarchy navigator, 26, 87
hierarchy priority, 216
hierarchy reports, 89

I

ID block
adding, 78
implicit connections, 120, 125
instances, 114
changing properties in all, 149
converting to different cell, 119
hiding, 117
locking, 117
naming constraints, 138
placing new, 117
preventing selection, 117
setting properties while placing, 156
validation, 138
viewing a specific one, 167
viewing in context, 44
interface
cell, 53, 56
examples, 57
interface view, 56
internal units, 39
interpolation
Fourier analysis, 224
interpreted properties, 148
iPar() function, 162

K

keyboard shortcuts
adding or changing, 18
keyword groups in text files, 49

L

L corner segment, 99
labels
drawing, 102
repeating, 103
language
changing, 44
libraries
adding, 59, 62, 85
cross-referencing, 59

list of used in design, 60
loading, 62, 205
location, 62
missing, 65, 67
netlisting, 213
redirected, 67
removing, 86
renaming, 69
saving, 69
unresolved, 65
libraries list, 205
Libraries navigator, 25, 84
libraries.list, 60, 62
setup files, 205
lines
drawing, 101
Locator toolbar, 11
lock files, 60
locking
cells, 43
instances, 117
log files
displaying, 30
filtering, 32
where saved, 30

M

menu bar, 10
mouse
scrolling with wheel, 45
mouse buttons
drawing mode, 99
Mouse buttons toolbar, 12
mouse snapping grid, 39, 40
moving objects, 109
using numeric values, 109

N

name validation, 138
naming constraints, 138
customizing, 138
default, 138
TCL functions for, 138
net caps, 134
net labels
repeating, 103
netlists
exporting, 182
TPR format, 198
nets, 114, 119, 126
capping, 134
connection points, 120
global, 123, 132, 133
hidden, 133
highlighting, 236
labeling, 125
naming, 119, 125
naming constraints, 138
placeholder, 133
validation, 138
node capping, 134, 192
node voltage
initial condition values, 223
node voltage values, 222
noise analysis, 227
non-exclusive access, 37, 43, 63
number of points, in Fourier analysis, 224

O

object types, 98
OnInstanceProc, 156
open.design, 205
origin point, 39
output properties, 149

P

PageID, 77
PageID cell, 78
pages
frames, 42
size, 42
panning, 83
to a coordinate, 83
to found objects, 47
with the mouse wheel, 45
parameter sweep, 228
parametric sweep, 228
parent cells, 86
pasting
objects, 112
work area as bitmap, 112
path name
evaluation, 61, 211
paths
drawing, 101
performance impact
temperature sweep, 228
pin, 22
pinned view, 22
pinned views, docked views, 22
polygons
drawing, 101
pop out of context
"pop context" in net highlighting, 140

"pop context" to view properties, 167
"pop context" while probing, 236
in schematic views, 44
port labels
repeating, 103
ports, 114, 122
adding, 123
annotating, 165
displaying voltage, charge or current, 165
generating for symbols, 145
generating from a netlist, 57
global, 123, 131, 173
graphics, 124
implicit connection, 120
labeling, 103
name checking, 145
naming, 124, 125
naming checking, 138
placement, 158
properties, 127
symbolic and schematic correspondence, 124
types, 123
validation, 138
pPar() function, 162
precise,in SPICE simulation, 211
primitive cells, 86
print start time
for transient analysis, 223
print time step
for transient analysis, 223
printing
page setup, 42
pages, 42
title or ID block, 78
title or ID blocks, 78
probe data
output file, 211
probe data file, 233
probing
data output file values, 211
data output files
device terminal current, 211
nets, 235
pins, 235
subcircuits, 235
to SPICE primitives, 236
voltage, current, charge, 234
probing buses, 236
product support, 51
product version
displaying, 51
project setup folder, 36
location, 36
properties, 114, 119
adding, 153
as expressions, 143
changing in all instances of a cell, 149
copying, 154, 208
default, 149
default values symbols, 143
defined, 147
editing from the Properties window, 151
editing from the work area, 118
editing in work area, 118
editing on multiple instances, 119
empty or zero value, 148
evaluated, 118
evaluating "$", 151
hidden, 144, 148
in SPICE statements, 143
inheriting, 116, 156
interpreted expressions, 148
locked, 118
moving, 118
netlisting, 185
OnInstanceProc, 156
operators for expressions, 160
outputting, 148
overriding, 116
overriding in instances, 143
pasting, 154, 208
renaming, 154
selecting from the work area, 118
selecting in the design window, 155
service, 149
settin g while instancing, 156
simulation output, 149
SPICE.DEFINITION, 185
SPICE.MODEL, 186
SPICE.ORDER, 185
SPICE.OUTPUT, 184, 190
SPICE.PARAMETERS, 187
SPICE.PINORDER, 186
SPICE.PREFIX, 186
SPICE.PRIMITIVE, 185
sub-properties, 149
system, 148
user, 148
value only, 144
visible, 144
Properties window, 27
property references, 162
push into context, 140, 236
in schematic views, 44
to view properties, 167

R

replace device, 154
replace symbol, 154
restoring an operation, 112

rotating objects, 110
rubberbanding, 109, 122

S

scaling
  for maximum grid size, 40
  setting for a design, 39
schematic view
  cells, 54
screentips, 20
scripts folder, 61
scrolling
  with the mouse wheel, 45
Segment toolbar, 11
segment types
  changing, 98
  drawing, 98
selecting objects
  cycling selection, 107
  explicit selection, 107
  extending selection, 107
  implicit selection, 108
selection
  behavior, 46
  explicit, 107
  extending, 107
selective evaluations, 163
service properties, 149
setup files
  libraries.list, 205
setup folder, 61
setup options
  by library, 37
  changing, 34, 37
  design checks, 136
  editing cell contents, 43
  keyword groups, 49
  language, 44
  loading, 9
  loading from TCL files, 34, 37
  mouse wheel, 45
  naming constraints, 138
  project setup folder, 36
  reusing design windows, 44
  saving, 9, 34
  saving local settings, 35
  selection behavior, 46
  SPICE simulation, 154, 208
  text file extensions, 49
  text file updates, 47
  user preferences folder, 36
shortcuts for viewing hierarchy, 87
simulation output
  file, 213
  properties, 149
simulation toolbar, 11
slow-right-click, 216
small-signal
  input source, 227
  variables, 227
small-signal parameters
  probing, 235
snap grid, 39, 40
  for solder points, 121
Solder dots, scaling, 41
solder points, 121
source scripts, 61
source sweeping, 224
SPICE command wizard, 218, 219
SPICE files, 168
  additional SPICE commands, 218
  controlling export order, 183
  exporting, 135, 180
  exporting as subcircuit, 182
  exporting as top level cell, 182
  exporting simulation parameters, 154, 208
  formats supported in S-Edit, 169
  importing simulation parameters, 154, 208
  netlists, exporting, 184
  options, 221
  output options, 183
  parameters, 221
  properties as expressions in, 143
  saving simulation setups, 154, 208
  translation for Cadence, 173
SPICE setting
  accurate, 211
  default, 211
  fast, 211
  precise, 211
SPICE Simulation toolbar, 11
SPICE view, 55
SPICE.DEFINITION, 185
SPICE.MODEL, 186
SPICE.ORDER, 185
SPICE.OUTPUT, 174, 184, 190
  exporting properties, 184
  in EDIF, 171
  properties, 184
SPICE.PARAMETERS, 187
SPICE.PINORDER, 186
SPICE.PREFIX, 186
SPICE.PRIMITIVE, 185
Standard toolbar, 11
startup folder, 138
Status bar, 22
step size, in sweep analysis, 225
stop time, for transient analysis, 223
subcircuit definition, 183, 185, 186, 187
  passing parameters down the hierarchy, 190

subcircuit parameter passing, 190
subcircuits
    exporting from netlist, 182
    probing, 235
support diagnostics, 52
sweep analysis, 224
symbol view, 54
symbols
    automatically updating, 145
    creating, 143
    generating from schematic views, 145
    global, 132
    properties for, 147
    replacing, 154
    views, switching schematics for, 59
system information
    displaying, 51
system properties, 78, 148

T

Tanner Place and Route format, 198
Tannerbetical sort order, 180
TCL commands
    available in S-Edit, 201
    help for, 200
TCL files
    built in functions, 163
    commands in properties, 163
    commands used in S-Edit, 201
    editing, 28, 65, 202
    executing, 65, 200, 202
    filtering events to log, 18, 32
    help for commands used in S-Edit, 200
    loading, 65
    log file, 29
    name validation functions, 138
    order of execution, 205
    running, 28
    running automatically, 204
    saving, 65
    saving and loading setups, 34, 37
    source scripts, 202
    writing, 200
TCL functions, 163
    for evaluating expressions, 163
temperature sweep, 228
text editor
    local and remote file updates, 47
text files, 61
    extentions, 49
    external updates, 47
    files formats, 49
    keyword groups, 49
    opening, 62
timestep
    for transient analysis, 223
title bar, 10
title block
    adding to a schematic page, 78
    preventing export, 194, 196
    printing, 77, 78
Tnom
    in SPICE simulation, 210, 215
toolbars
    deleting, 12
    drawing, 11
    electrical, 11
    locator, 11
    mouse buttons, 12
    renaming, 12
    resetting to default display, 12
    segment, 11
    simulation, 11
    standard, 11
    status, 22
tooltips, 20
top-level cells, 86
TPR format, 198
TPR netlists
    exporting, 197
transfer function analysis, 227
Transient analysis, 222
transient simulation
    startup mode, 223

U

undoing an operation, 112
unnamed, default naming, 126
unresolved cells, 65
unresolved libraries, 65
user preferences folder, 36, 205
    location, 36
user properties, 148
    adding, 153
    copying, 154, 208
    pasting, 154, 208
    renaming, 154

V

validation
    of names, 138
values in probe data output file, 211
vectors
    repeating bits in a vector term, 128
verbosity, 32, 203
Verilog

.DEFINITION property, 194
.MODEL property, 194
.OUTPUT property, 194
.PRIMITIVE property, 194
export properties, 194
files, 168
omitting global parameters during export, 194
output options, 194
Verilog-A view, 55
Verilog files
exporting, 135, 180
VHDL
output options, 196
VHDL files, 168
design checks on export, 195
exporting, 135, 180, 196
view type
interface, 56
view types
examples, 57
interface, 53, 56
setting priority for export and netlisting, 216
view types, Verilog
Verilog-A view type, 216
ViewDraw
properties ORDER, PINORDER, PREFIX, 171
views
copying, 73
creating, 71, 115
deleting, 75
naming constraints, 138
opening, 70, 71
pinned, 22
renaming, 75
types of, 54, 55
validation, 138
Vista operating system
startup script locations, 204
voltage source modules, 133

W

waveform display
controlling, 211
waveform display, controlling, 233
waveform probing, 233, 234
device terminals, 235
nets, 235
probe data file, 233
properties in context, 140, 236
subcircuits, 235
with .probe in netlist, 233
with SPICE OUTPUT property, 233
wires
adding a conection point, 121
attaching,detaching, 122
connecting, 122
disconnecting, 122
drawing, 101, 104, 120
implicit connections, 120, 125
rubberbanding, 122
split automatically, 122
work area, 22, 80
panning, 83
work space settings, 60
write reservations, 37, 43, 63

Z

zooming, 83
to found objects, 47
to selected objects, 84
to show all objects, 84
with the keyboard, 84
with the mouse, 84
with the mouse wheel, 45, 84

Credits

Software Development

Seyran Avanesyan
Barry Dyne
Dan'l Leviton
Kurt Van Ness
David Cardoze
Nikita Jorniak
Massimo Sivilotti
Nicolas Williams

Quality Assurance

Luba Gromova

Documentation

Judy Bergstresser
Ken Van de Houten
Barry Dyne

End-User License Agreement

The latest version of the End-User License Agreement is available on-line at:
www.mentor.com/eula

IMPORTANT INFORMATION

USE OF ALL SOFTWARE IS SUBJECT TO LICENSE RESTRICTIONS. CAREFULLY READ THIS LICENSE AGREEMENT BEFORE USING THE PRODUCTS. USE OF SOFTWARE INDICATES CUSTOMER'S COMPLETE AND UNCONDITIONAL ACCEPTANCE OF THE TERMS AND CONDITIONS SET FORTH IN THIS AGREEMENT. ANY ADDITIONAL OR DIFFERENT PURCHASE ORDER TERMS AND CONDITIONS SHALL NOT APPLY.

END-USER LICENSE AGREEMENT ("Agreement")

This is a legal agreement concerning the use of Software (as defined in Section 2) and hardware (collectively "Products") between the company acquiring the Products ("Customer"), and the Mentor Graphics entity that issued the corresponding quotation or, if no quotation was issued, the applicable local Mentor Graphics entity ("Mentor Graphics"). Except for license agreements related to the subject matter of this license agreement which are physically signed by Customer and an authorized representative of Mentor Graphics, this Agreement and the applicable quotation contain the parties' entire understanding relating to the subject matter and supersede all prior or contemporaneous agreements. If Customer does not agree to these terms and conditions, promptly return or, in the case of Software received electronically, certify destruction of Software and all accompanying items within five days after receipt of Software and receive a full refund of any license fee paid.

1. **ORDERS, FEES AND PAYMENT.**

   1.1. To the extent Customer (or if agreed by Mentor Graphics, Customer's appointed third party buying agent) places and Mentor Graphics accepts purchase orders pursuant to this Agreement (each an "Order"), each Order will constitute a contract between Customer and Mentor Graphics, which shall be governed solely and exclusively by the terms and conditions of this Agreement, any applicable addenda and the applicable quotation, whether or not those documents are referenced on the Order. Any additional or conflicting terms and conditions appearing on an Order or presented in any electronic portal or automated order management system, whether or not required to be electronically accepted, will not be effective unless agreed in writing and physically signed by an authorized representative of Customer and Mentor Graphics.

   1.2. Amounts invoiced will be paid, in the currency specified on the applicable invoice, within 30 days from the date of such invoice. Any past due invoices will be subject to the imposition of interest charges in the amount of one and one-half percent per month or the applicable legal rate currently in effect, whichever is lower. Prices do not include freight, insurance, customs duties, taxes or other similar charges, which Mentor Graphics will state separately in the applicable invoice. Unless timely provided with a valid certificate of exemption or other evidence that items are not taxable, Mentor Graphics will invoice Customer for all applicable taxes including, but not limited to, VAT, GST, sales tax, consumption tax and service tax. Customer will make all payments free and clear of, and without reduction for, any withholding or other taxes; any such taxes imposed on payments by Customer hereunder will be Customer's sole responsibility. If Customer appoints a third party to place purchase orders and/or make payments on Customer's behalf, Customer shall be liable for payment under Orders placed by such third party in the event of default.

   1.3. All Products are delivered FCA factory (Incoterms 2010), freight prepaid and invoiced to Customer, except Software delivered electronically, which shall be deemed delivered when made available to Customer for download. Mentor Graphics retains a security interest in all Products delivered under this Agreement, to secure payment of the purchase price of such Products, and Customer agrees to sign any documents that Mentor Graphics determines to be necessary or convenient for use in filing or perfecting such security interest. Mentor Graphics' delivery of Software by electronic means is subject to Customer's provision of both a primary and an alternate e-mail address.

2. **GRANT OF LICENSE.** The software installed, downloaded, or otherwise acquired by Customer under this Agreement, including any updates, modifications, revisions, copies, documentation and design data ("Software") are copyrighted, trade secret and confidential information of Mentor Graphics or its licensors, who maintain exclusive title to all Software and retain all rights not expressly granted by this Agreement. Mentor Graphics grants to Customer, subject to payment of applicable license fees, a nontransferable, nonexclusive license to use Software solely: (a) in machine-readable, object-code form (except as provided in Subsection 5.2); (b) for Customer's internal business purposes; (c) for the term of the license; and (d) on the computer hardware and at the site authorized by Mentor Graphics. A site is restricted to a one-half mile (800 meter) radius. Customer may have Software temporarily used by an employee for telecommuting purposes from locations other than a Customer office, such as the employee's residence, an airport or hotel, provided that such employee's primary place of employment is the site where the Software is authorized for use. Mentor Graphics' standard policies and programs, which vary depending on Software, license fees paid or services purchased, apply to the following: (a) relocation of Software; (b) use of Software, which may be limited, for example, to execution of a single session by a single user on the authorized hardware or for a restricted period of time (such limitations may be technically implemented through the use of authorization codes or similar devices); and (c) support services provided, including eligibility to receive telephone support, updates, modifications, and revisions. For the avoidance of doubt, if Customer provides any feedback or requests any change or enhancement to Products, whether in the course of receiving support or consulting services, evaluating Products, performing beta testing or otherwise, any inventions, product improvements, modifications or developments made by Mentor Graphics (at Mentor Graphics' sole discretion) will be the exclusive property of Mentor Graphics.

3. **ESC SOFTWARE.** If Customer purchases a license to use development or prototyping tools of Mentor Graphics' Embedded Software Channel ("ESC"), Mentor Graphics grants to Customer a nontransferable, nonexclusive license to reproduce and distribute executable

files created using ESC compilers, including the ESC run-time libraries distributed with ESC C and C++ compiler Software that are linked into a composite program as an integral part of Customer's compiled computer program, provided that Customer distributes these files only in conjunction with Customer's compiled computer program. Mentor Graphics does NOT grant Customer any right to duplicate, incorporate or embed copies of Mentor Graphics' real-time operating systems or other embedded software products into Customer's products or applications without first signing or otherwise agreeing to a separate agreement with Mentor Graphics for such purpose.

4. **BETA CODE.**

4.1. Portions or all of certain Software may contain code for experimental testing and evaluation (which may be either alpha or beta, collectively "Beta Code"), which may not be used without Mentor Graphics' explicit authorization. Upon Mentor Graphics' authorization, Mentor Graphics grants to Customer a temporary, nontransferable, nonexclusive license for experimental use to test and evaluate the Beta Code without charge for a limited period of time specified by Mentor Graphics. Mentor Graphics may choose, at its sole discretion, not to release Beta Code commercially in any form.

4.2. If Mentor Graphics authorizes Customer to use the Beta Code, Customer agrees to evaluate and test the Beta Code under normal conditions as directed by Mentor Graphics. Customer will contact Mentor Graphics periodically during Customer's use of the Beta Code to discuss any malfunctions or suggested improvements. Upon completion of Customer's evaluation and testing, Customer will send to Mentor Graphics a written evaluation of the Beta Code, including its strengths, weaknesses and recommended improvements.

4.3. Customer agrees to maintain Beta Code in confidence and shall restrict access to the Beta Code, including the methods and concepts utilized therein, solely to those employees and Customer location(s) authorized by Mentor Graphics to perform beta testing. Customer agrees that any written evaluations and all inventions, product improvements, modifications or developments that Mentor Graphics conceived or made during or subsequent to this Agreement, including those based partly or wholly on Customer's feedback, will be the exclusive property of Mentor Graphics. Mentor Graphics will have exclusive rights, title and interest in all such property. The provisions of this Subsection 4.3 shall survive termination of this Agreement.

5. **RESTRICTIONS ON USE.**

5.1. Customer may copy Software only as reasonably necessary to support the authorized use. Each copy must include all notices and legends embedded in Software and affixed to its medium and container as received from Mentor Graphics. All copies shall remain the property of Mentor Graphics or its licensors. Customer shall maintain a record of the number and primary location of all copies of Software, including copies merged with other software, and shall make those records available to Mentor Graphics upon request. Customer shall not make Products available in any form to any person other than Customer's employees and on-site contractors, excluding Mentor Graphics competitors, whose job performance requires access and who are under obligations of confidentiality. Customer shall take appropriate action to protect the confidentiality of Products and ensure that any person permitted access does not disclose or use Products except as permitted by this Agreement. Customer shall give Mentor Graphics written notice of any unauthorized disclosure or use of the Products as soon as Customer becomes aware of such unauthorized disclosure or use. Except as otherwise permitted for purposes of interoperability as specified by applicable and mandatory local law, Customer shall not reverse-assemble, reverse-compile, reverse-engineer or in any way derive any source code from Software. Log files, data files, rule files and script files generated by or for the Software (collectively "Files"), including without limitation files containing Standard Verification Rule Format ("SVRF") and Tcl Verification Format ("TVF") which are Mentor Graphics' trade secret and proprietary syntaxes for expressing process rules, constitute or include confidential information of Mentor Graphics. Customer may share Files with third parties, excluding Mentor Graphics competitors, provided that the confidentiality of such Files is protected by written agreement at least as well as Customer protects other information of a similar nature or importance, but in any case with at least reasonable care. Customer may use Files containing SVRF or TVF only with Mentor Graphics products. Under no circumstances shall Customer use Products or Files or allow their use for the purpose of developing, enhancing or marketing any product that is in any way competitive with Products, or disclose to any third party the results of, or information pertaining to, any benchmark.

5.2. If any Software or portions thereof are provided in source code form, Customer will use the source code only to correct software errors and enhance or modify the Software for the authorized use. Customer shall not disclose or permit disclosure of source code, in whole or in part, including any of its methods or concepts, to anyone except Customer's employees or on-site contractors, excluding Mentor Graphics competitors, with a need to know. Customer shall not copy or compile source code in any manner except to support this authorized use.

5.3. Customer may not assign this Agreement or the rights and duties under it, or relocate, sublicense, or otherwise transfer the Products, whether by operation of law or otherwise ("Attempted Transfer"), without Mentor Graphics' prior written consent and payment of Mentor Graphics' then-current applicable relocation and/or transfer fees. Any Attempted Transfer without Mentor Graphics' prior written consent shall be a material breach of this Agreement and may, at Mentor Graphics' option, result in the immediate termination of the Agreement and/or the licenses granted under this Agreement. The terms of this Agreement, including without limitation the licensing and assignment provisions, shall be binding upon Customer's permitted successors in interest and assigns.

5.4. The provisions of this Section 5 shall survive the termination of this Agreement.

6. **SUPPORT SERVICES.** To the extent Customer purchases support services, Mentor Graphics will provide Customer with updates and technical support for the Products, at the Customer site(s) for which support is purchased, in accordance with Mentor Graphics' then current End-User Support Terms located at http://supportnet.mentor.com/supportterms.

7. **LIMITED WARRANTY.**

7.1. Mentor Graphics warrants that during the warranty period its standard, generally supported Products, when properly installed, will substantially conform to the functional specifications set forth in the applicable user manual. Mentor Graphics does not warrant that Products will meet Customer's requirements or that operation of Products will be uninterrupted or error free. The

warranty period is 90 days starting on the 15th day after delivery or upon installation, whichever first occurs. Customer must notify Mentor Graphics in writing of any nonconformity within the warranty period. For the avoidance of doubt, this warranty applies only to the initial shipment of Software under an Order and does not renew or reset, for example, with the delivery of (a) Software updates or (b) authorization codes or alternate Software under a transaction involving Software re-mix. This warranty shall not be valid if Products have been subject to misuse, unauthorized modification, improper installation or Customer is not in compliance with this Agreement. MENTOR GRAPHICS' ENTIRE LIABILITY AND CUSTOMER'S EXCLUSIVE REMEDY SHALL BE, AT MENTOR GRAPHICS' OPTION, EITHER (A) REFUND OF THE PRICE PAID UPON RETURN OF THE PRODUCTS TO MENTOR GRAPHICS OR (B) MODIFICATION OR REPLACEMENT OF THE PRODUCTS THAT DO NOT MEET THIS LIMITED WARRANTY. MENTOR GRAPHICS MAKES NO WARRANTIES WITH RESPECT TO: (A) SERVICES; (B) PRODUCTS PROVIDED AT NO CHARGE; OR (C) BETA CODE; ALL OF WHICH ARE PROVIDED "AS IS."

7.2. THE WARRANTIES SET FORTH IN THIS SECTION 7 ARE EXCLUSIVE. NEITHER MENTOR GRAPHICS NOR ITS LICENSORS MAKE ANY OTHER WARRANTIES EXPRESS, IMPLIED OR STATUTORY, WITH RESPECT TO PRODUCTS PROVIDED UNDER THIS AGREEMENT. MENTOR GRAPHICS AND ITS LICENSORS SPECIFICALLY DISCLAIM ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY.

8. **LIMITATION OF LIABILITY.** EXCEPT WHERE THIS EXCLUSION OR RESTRICTION OF LIABILITY WOULD BE VOID OR INEFFECTIVE UNDER APPLICABLE LAW, IN NO EVENT SHALL MENTOR GRAPHICS OR ITS LICENSORS BE LIABLE FOR INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES (INCLUDING LOST PROFITS OR SAVINGS) WHETHER BASED ON CONTRACT, TORT OR ANY OTHER LEGAL THEORY, EVEN IF MENTOR GRAPHICS OR ITS LICENSORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT SHALL MENTOR GRAPHICS' OR ITS LICENSORS' LIABILITY UNDER THIS AGREEMENT EXCEED THE AMOUNT RECEIVED FROM CUSTOMER FOR THE HARDWARE, SOFTWARE LICENSE OR SERVICE GIVING RISE TO THE CLAIM. IN THE CASE WHERE NO AMOUNT WAS PAID, MENTOR GRAPHICS AND ITS LICENSORS SHALL HAVE NO LIABILITY FOR ANY DAMAGES WHATSOEVER. THE PROVISIONS OF THIS SECTION 8 SHALL SURVIVE THE TERMINATION OF THIS AGREEMENT.

9. **HAZARDOUS APPLICATIONS.** CUSTOMER ACKNOWLEDGES IT IS SOLELY RESPONSIBLE FOR TESTING ITS PRODUCTS USED IN APPLICATIONS WHERE THE FAILURE OR INACCURACY OF ITS PRODUCTS MIGHT RESULT IN DEATH OR PERSONAL INJURY ("HAZARDOUS APPLICATIONS"). EXCEPT TO THE EXTENT THIS EXCLUSION OR RESTRICTION OF LIABILITY WOULD BE VOID OR INEFFECTIVE UNDER APPLICABLE LAW, IN NO EVENT SHALL MENTOR GRAPHICS OR ITS LICENSORS BE LIABLE FOR ANY DAMAGES RESULTING FROM OR IN CONNECTION WITH THE USE OF MENTOR GRAPHICS PRODUCTS IN OR FOR HAZARDOUS APPLICATIONS. THE PROVISIONS OF THIS SECTION 9 SHALL SURVIVE THE TERMINATION OF THIS AGREEMENT.

10. **INDEMNIFICATION.** CUSTOMER AGREES TO INDEMNIFY AND HOLD HARMLESS MENTOR GRAPHICS AND ITS LICENSORS FROM ANY CLAIMS, LOSS, COST, DAMAGE, EXPENSE OR LIABILITY, INCLUDING ATTORNEYS' FEES, ARISING OUT OF OR IN CONNECTION WITH THE USE OF MENTOR GRAPHICS PRODUCTS IN OR FOR HAZARDOUS APPLICATIONS. THE PROVISIONS OF THIS SECTION 10 SHALL SURVIVE THE TERMINATION OF THIS AGREEMENT.

11. **INFRINGEMENT.**

11.1. Mentor Graphics will defend or settle, at its option and expense, any action brought against Customer in the United States, Canada, Japan, or member state of the European Union which alleges that any standard, generally supported Product acquired by Customer hereunder infringes a patent or copyright or misappropriates a trade secret in such jurisdiction. Mentor Graphics will pay costs and damages finally awarded against Customer that are attributable to such action. Customer understands and agrees that as conditions to Mentor Graphics' obligations under this section Customer must: (a) notify Mentor Graphics promptly in writing of the action; (b) provide Mentor Graphics all reasonable information and assistance to settle or defend the action; and (c) grant Mentor Graphics sole authority and control of the defense or settlement of the action.

11.2. If a claim is made under Subsection 11.1 Mentor Graphics may, at its option and expense: (a) replace or modify the Product so that it becomes noninfringing; (b) procure for Customer the right to continue using the Product; or (c) require the return of the Product and refund to Customer any purchase price or license fee paid, less a reasonable allowance for use.

11.3. Mentor Graphics has no liability to Customer if the action is based upon: (a) the combination of Software or hardware with any product not furnished by Mentor Graphics; (b) the modification of the Product other than by Mentor Graphics; (c) the use of other than a current unaltered release of Software; (d) the use of the Product as part of an infringing process; (e) a product that Customer makes, uses, or sells; (f) any Beta Code or Product provided at no charge; (g) any software provided by Mentor Graphics' licensors who do not provide such indemnification to Mentor Graphics' customers; or (h) infringement by Customer that is deemed willful. In the case of (h), Customer shall reimburse Mentor Graphics for its reasonable attorney fees and other costs related to the action.

11.4. THIS SECTION 11 IS SUBJECT TO SECTION 8 ABOVE AND STATES THE ENTIRE LIABILITY OF MENTOR GRAPHICS AND ITS LICENSORS, AND CUSTOMER'S SOLE AND EXCLUSIVE REMEDY, FOR DEFENSE, SETTLEMENT AND DAMAGES, WITH RESPECT TO ANY ALLEGED PATENT OR COPYRIGHT INFRINGEMENT OR TRADE SECRET MISAPPROPRIATION BY ANY PRODUCT PROVIDED UNDER THIS AGREEMENT.

12. **TERMINATION AND EFFECT OF TERMINATION.**

12.1. If a Software license was provided for limited term use, such license will automatically terminate at the end of the authorized term. Mentor Graphics may terminate this Agreement and/or any license granted under this Agreement immediately upon written notice if Customer: (a) exceeds the scope of the license or otherwise fails to comply with the licensing or confidentiality provisions of this Agreement, or (b) becomes insolvent, files a bankruptcy petition, institutes proceedings for liquidation or winding up or enters into an agreement to assign its assets for the benefit of creditors. For any other material breach of any

provision of this Agreement, Mentor Graphics may terminate this Agreement and/or any license granted under this Agreement upon 30 days written notice if Customer fails to cure the breach within the 30 day notice period. Termination of this Agreement or any license granted hereunder will not affect Customer's obligation to pay for Products shipped or licenses granted prior to the termination, which amounts shall be payable immediately upon the date of termination.

12.2. Upon termination of this Agreement, the rights and obligations of the parties shall cease except as expressly set forth in this Agreement. Upon termination, Customer shall ensure that all use of the affected Products ceases, and shall return hardware and either return to Mentor Graphics or destroy Software in Customer's possession, including all copies and documentation, and certify in writing to Mentor Graphics within ten business days of the termination date that Customer no longer possesses any of the affected Products or copies of Software in any form.

13. **EXPORT.** The Products provided hereunder are subject to regulation by local laws and United States ("U.S.") government agencies, which prohibit export, re-export or diversion of certain products, information about the products, and direct or indirect products thereof, to certain countries and certain persons. Customer agrees that it will not export or re-export Products in any manner without first obtaining all necessary approval from appropriate local and U.S. government agencies. If Customer wishes to disclose any information to Mentor Graphics that is subject to any U.S. or other applicable export restrictions, including without limitation the U.S. International Traffic in Arms Regulations (ITAR) or special controls under the Export Administration Regulations (EAR), Customer will notify Mentor Graphics personnel, in advance of each instance of disclosure, that such information is subject to such export restrictions.

14. **U.S. GOVERNMENT LICENSE RIGHTS.** Software was developed entirely at private expense. The parties agree that all Software is commercial computer software within the meaning of the applicable acquisition regulations. Accordingly, pursuant to U.S. FAR 48 CFR 12.212 and DFAR 48 CFR 227.7202, use, duplication and disclosure of the Software by or for the U.S. government or a U.S. government subcontractor is subject solely to the terms and conditions set forth in this Agreement, which shall supersede any conflicting terms or conditions in any government order document, except for provisions which are contrary to applicable mandatory federal laws.

15. **THIRD PARTY BENEFICIARY.** Mentor Graphics Corporation, Mentor Graphics (Ireland) Limited, Microsoft Corporation and other licensors may be third party beneficiaries of this Agreement with the right to enforce the obligations set forth herein.

16. **REVIEW OF LICENSE USAGE.** Customer will monitor the access to and use of Software. With prior written notice and during Customer's normal business hours, Mentor Graphics may engage an internationally recognized accounting firm to review Customer's software monitoring system and records deemed relevant by the internationally recognized accounting firm to confirm Customer's compliance with the terms of this Agreement or U.S. or other local export laws. Such review may include FlexNet (or successor product) report log files that Customer shall capture and provide at Mentor Graphics' request. Customer shall make records available in electronic format and shall fully cooperate with data gathering to support the license review. Mentor Graphics shall bear the expense of any such review unless a material non-compliance is revealed. Mentor Graphics shall treat as confidential information all information gained as a result of any request or review and shall only use or disclose such information as required by law or to enforce its rights under this Agreement. The provisions of this Section 16 shall survive the termination of this Agreement.

17. **CONTROLLING LAW, JURISDICTION AND DISPUTE RESOLUTION.** The owners of certain Mentor Graphics intellectual property licensed under this Agreement are located in Ireland and the U.S. To promote consistency around the world, disputes shall be resolved as follows: excluding conflict of laws rules, this Agreement shall be governed by and construed under the laws of the State of Oregon, U.S., if Customer is located in North or South America, and the laws of Ireland if Customer is located outside of North or South America. All disputes arising out of or in relation to this Agreement shall be submitted to the exclusive jurisdiction of the courts of Portland, Oregon when the laws of Oregon apply, or Dublin, Ireland when the laws of Ireland apply. Notwithstanding the foregoing, all disputes in Asia arising out of or in relation to this Agreement shall be resolved by arbitration in Singapore before a single arbitrator to be appointed by the chairman of the Singapore International Arbitration Centre ("SIAC") to be conducted in the English language, in accordance with the Arbitration Rules of the SIAC in effect at the time of the dispute, which rules are deemed to be incorporated by reference in this section. Nothing in this section shall restrict Mentor Graphics' right to bring an action (including for example a motion for injunctive relief) against Customer in the jurisdiction where Customer's place of business is located. The United Nations Convention on Contracts for the International Sale of Goods does not apply to this Agreement.

18. **SEVERABILITY.** If any provision of this Agreement is held by a court of competent jurisdiction to be void, invalid, unenforceable or illegal, such provision shall be severed from this Agreement and the remaining provisions will remain in full force and effect.

19. **MISCELLANEOUS.** This Agreement contains the parties' entire understanding relating to its subject matter and supersedes all prior or contemporaneous agreements. Some Software may contain code distributed under a third party license agreement that may provide additional rights to Customer. Please see the applicable Software documentation for details. This Agreement may only be modified in writing, signed by an authorized representative of each party. Waiver of terms or excuse of breach must be in writing and shall not constitute subsequent consent, waiver or excuse.

Rev. 140201, Part No. 258976